UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA, MANAGUA

UNAN - MANAGUA

FACULTAD REGIONAL MULTIDISCIPLINARIA DE CHONTALES

FAREM-CHONTALES

PROGRAMA DE DOCTORADO EN MATEMATICA APLICADA

Tesis Para Optar Al Grado De Doctor En Matemática Aplicada

Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica

Autor:M.Sc. Rudy Alberto López Potosme

Tutor:M.Sc.-Ph.D. Antonio Parajón Guevara

Noviembre, 2017

ANTONIO PARAJÓN GUEVARA, profesor titular del Departamento de Matemática de la Facultad de Educación e Idiomas de la Universidad Nacional Autónoma de Nicaragua, UNAN-Managua.

CERTIFICA que la presente memoria de investigación:

Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica

Ha sido realizada bajo su dirección en el PROGRAMA DE DOCTORADO EN MATEMÁTICA APLICADA por el Máster **Rudy Alberto López Potosme**, y constituye su tesis para optar al grado de Doctor en Matemática Aplicada.

Y para que así conste, en cumplimiento con la normativa vigente de posgrado, autoriza su presentación ante la Facultad Regional Multidisciplinaria De Chontales (FAREM-Chontales) para que pueda ser tramitada su lectura y Defensa pública.

Managua, Nicaragua, 21 de Noviembre de 2017.

El DIRECTOR DE LA TESIS

Antonio Parajón Guevara, MS.c - Ph.D

DEDICATORIA

A mi familia y amigos, por todo su apoyo.

AGRADECIMIENTOS

Gracias Señor... por llegar hasta el final.

A Dios, por sus bendiciones.

Al Dr. Antonio Parajón, por sus aportes, su academia y su amistad.

A los Maestros Marlon Díaz y Hugo Gutiérrez por su confianza y apoyo.

A los profesores cubanos, por su grandiosa colaboración en el desarrollo del Programa de Doctorado en Matemática Aplicada.

A mis compañeros y amigos, por su colaboración.

Rudy Alberto López Postome.
Managua, Nicaragua. 21 de Noviembre de 2017.

TRAYECTORIA ACADÉMICA DEL DOCENTE INVESTIGADOR

Rudy Alberto López Potosme, es Licenciado en Ciencias de la Educación con Mención en Matemática por la Universidad Nacional Autónoma de Nicaragua, Managua (2004). Es Máster en Formación de Formadores por la Universidad Nacional Autónoma de Nicaragua, Managua (2009). Es Máster en Matemática Aplicada por la Universidad Nacional Autónoma de Nicaragua, Managua (2017). Es Docente Titular con Maestría en la UNAN-Managua, trabajando desde el 2004 en las áreas de Didáctica de Matemática, Estadística e Investigación de Operaciones, Análisis Matemático. Realizó estudios de Doctorado en Matemática Aplicada coordinado por la Universidad Central Marta Abreu de Las Villas, de la hermana república de Cuba. Realizó Tesis Doctoral, en Investigación de Operaciones, bajo la tutoría del Dr. Antonio Parajón Guevara. Línea de investigación sobre problemas de Optimización Combinatoria.

RESUMEN

Los problemas de Optimización Combinatoria aparecen en diversos contextos, como la distribución de carga física o eléctrica, detección de patrones de corte de piezas, redes de tráfico o telecomunicaciones, horarios de transportes laborales y escolares, fabricación de circuitos electrónicos, secuenciación de actividades en una empresa, entre otros. Esto hace que dichos problemas sean actrativos para estudiar ya sea desde el punto de vista teórico o práctico.

El propósito fundamental del trabajo fue la construcción de una metodología para abordar los problemas de Optimización Combinatoria, particularmente los siguientes: el Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de Piezas y el Problema del Agente Viajero. Cabe mencionar que esta metodología se caracteriza por resolver de una forma intuitiva, sencilla y práctica los problemas antes mencionados, utilizando algoritmos exactos y heurísticos, además de la implementación de la herramienta compuntacional WinQsb.

En lo que respecta al camino seguido para la realización del trabajo, es posible señalar que primeramente se realizó la construcción de una reseña histórica con la cual es posible comprender el génesis, desarrollo y el nivel de aplicación que tiene la Investigación de Operaciones, y en particular de los problemas de Optimización Combinatoria. Posteriormente para la elaboración de la metodología de solución que se propone, se revisó de forma exhaustiva el estado del arte de los modelos matemáticos, y de las diferentes técnicas de solución de los problemas en cuestión. Además de esto se realizó una revisión de las formas en como libros y otros documentos abordan estos problemas.

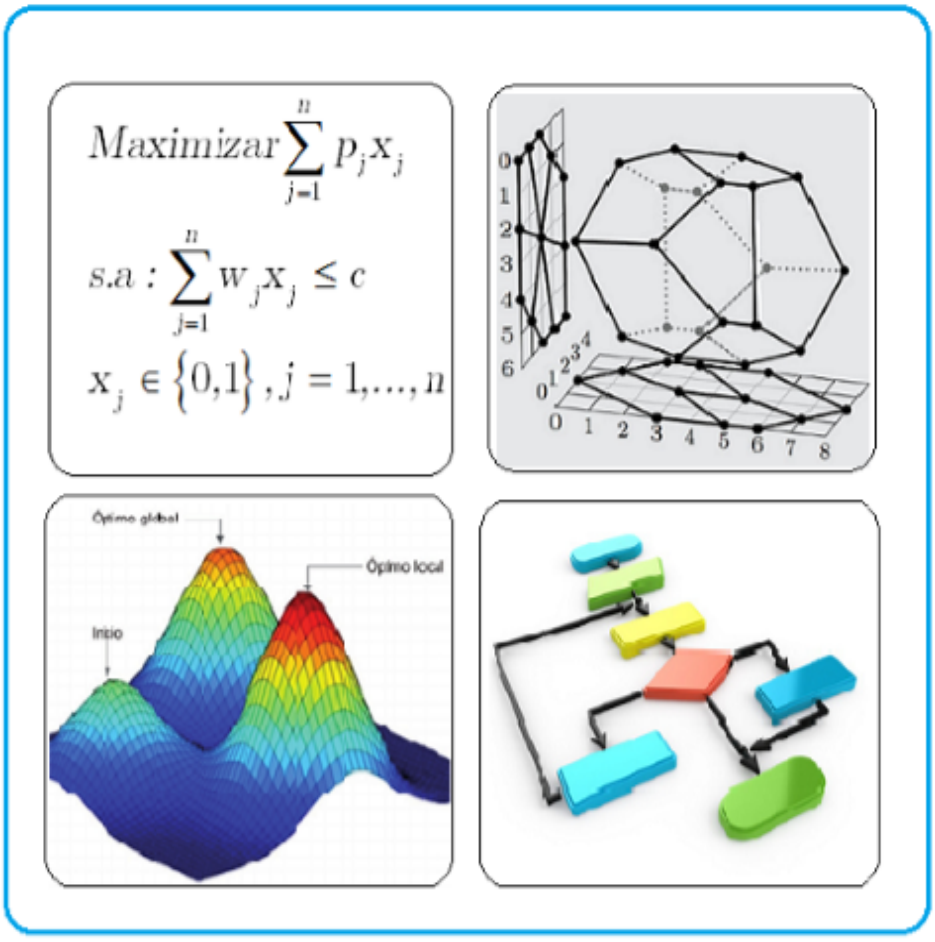
La forma en como resolver o tratar cada problema de Optimización Combinatoria se estructuró de la misma manera: a) Introducción, aquí se trata de dejar claro el problema, una breve reseña histórica, aplicaciones del problema y algunos algoritmos para resolverlo; b) Formulación matemática del problema, se muestra la función objetivo y las restricciones del modelo; c) Tratamiento metodológico, se resuelven ciertos problemas mediante la implementación de diversos algoritmos; seguidamente se presentan d) Problemas propuestos; e) Problemas resueltos y f) Bibliografía,la cual se decidió elaborarla para cada problema, dada la naturaleza heterogénea de los mismos.

Palabras claves: Investigación de Operaciones, Optimización Combinatoria, algoritmos exactos y heurísticos.

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN



En lenguaje coloquial optimizar es equivalente a mejorar, y cuando hablamos de mejoras nos referimos a aquellas que se hacen evidente en los atributos de las situaciones, por ejemplo cuando vamos de compras al supermercado queremos optimizar nuestro dinero, lo que significa adquirir la mayor cantidad de productos a un menor precio. Este término simple y sencillo podemos aplicarlo a muchos entornos, por ejemplo, en el día a día, hablamos de optimizar el espacio dentro de algo, optimizar los costos, optimizar el trabajo que realizamos a diario; en actividades industriales, por ejemplo se habla de maximizar la producción, minimizar los costos, optimizar la energía, optimizar la fuerza de trabajo y el tiempo, etc. Por lo que en realidad cuando hablamos de optimización nos referimos al propósito de hacer más con menos.

Particularmente en el contexto matemático, optimizar es el proceso de tratar de encontrar la mejor solución posible para un problema determinado. Es decir que el objetivo de un problema de Optimización es encontrar la solución óptima dado un criterio para discriminar entre dos soluciones. Es decir, se trata de encontrar el valor de las variables de decisión (sujetas a restricciones) para los que una determinada función objetivo alcanza su valor máximo o mínimo.

Cabe señalar que los problemas de Optimización en los que las variables de decisión son enteras, es decir, donde el espacio de soluciones está formado por ordenaciones o subconjuntos de números naturales, reciben el nombre de Optimización Combinatoria. En este caso, se trata de hallar el mejor valor de entre un número finito o numerable de soluciones viables. Sin embargo la enumeración de este conjunto resulta prácticamente intratable, aún para problemas de tamaño moderado.

Una gran cantidad y diversidad de problemas de optimización de recursos, ligados a situaciones reales, pueden ser formulados como problemas de Optimización Combinatoria: el ruteo y carga de vehículos en redes de distribución, el diseño de redes de telecomunicación, la planificación de la producción, la selección de carteras financieras, la asignación de tareas a procesadores, la asignación de tripulaciones en líneas aéreas, la planificación de la generación de electricidad y la distribución de ambulancias en una región para asegurar un cierto nivel de servicio a su población.

La mayoría de estos son problemas muy complejos de resolver, sin embargo, la necesidad práctica de su resolución, en forma exacta o aproximada, ha dado un gran impulso al estudio estructural de los mismos. Dada esta importancia y la necesidad que implica abordar dichos problemas, en este trabajo se expone una metodología, una forma alterna de trabajar algunos problemas de Optimización Combinatoria.

1.1 Tema delimitado

El presente trabajo versa sobre "*Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica*". Con el desarrollo del mismo se tratan de resolver de una forma intuitiva, sencilla y práctica los siguientes problemas clásicos de Optimización Combinatoria: el Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Problema del Agente Viajero (Travelling Salesman Problem). Es decir que la idea fundamental radica en proponer una metodología de solución para los problemas de Optimización Combinatoria antes mencionados, cuando tenemos asosciados a ellos, instancias de baja dimensionalidad.

Cada uno de estos problemas de Optimización Combinatoria se abordan siguiendo un mismo esquema central: introducción, formulación matemática del problema, tratamiento metodológico, problemas propuestos y su solución correspondiente, finalmente se presenta las referencias bibliográficas. En la introducción se trata de dejar claro de qué trata el problema, se expone una breve reseña histórica, algunas aplicaciones del problema, además se comenta acerca de las diversas técnicas resolución; en la formulación matemática se explica cuál es la función objetivo y las restricciones correspondientes; en el tratamiento metodológico se resuelven algunos problemas de baja dimensionalidad mediante la implementación de diversos algoritmos; seguidamente se propone una colección de problemas y luego se resuleven con almenos uno de los algoritmos previamente ejecutado; en el caso de la bibliografía, esta se decidió elaborarla para cada problema, dada la naturaleza heterogénea de los mismos.

De acuerdo con el reglamento del Sistema de Estudios de Posgrado y Educación Continua de La Universidad Nacional Autónoma de Nicaragua, Managua (2011). El área de conocimiento en que se circunscribe el presente proyecto de investigación es el Área de Ciencias Exactas, Ingeniería y Tecnología. Esto obedece a la naturaleza del estudio, en donde se abordan métodos y herramientas generales de Optimización Combinatoria, la cual tiene numerosas aplicaciones a problemas que se presentan en la industria, logística, ciencias, ingenierías y en la administración de organizaciones.

1.2 Planteamiento del problema

Una de las principales funciones de la universidad consiste en responder a las necesidades de una sociedad en continuo cambio. Estas necesidades deben ser retomadas en cada uno de los diferentes niveles de concreción del currículo universitario, y principalmente en el espacio donde se desarrolla el proceso de enseñanza y aprendizaje, esto con la finalidad de la generación de conocimiento útil para la vida individual y social tendiente a la elevación del nivel de conciencia propio de la sociedad creativa.

En este sentido, en las universidades, el proceso de enseñanza y aprendizaje, debería tener como propósito fundamental, promover la construcción de saberes que tengan significado y relevancia en la solución de problemas reales y cotidianos. Sin embargo para alcanzar dicho propósito es necesario la realización de transformaciones curriculares de forma continua y periódica.

Particularmente, en las Facultades donde se ofertan carreras de Ingeniería, Ciencias Exactas, Ciencias económicas, esta necesidad de satisfacer las demandas de la sociedad se manifiesta en los documentos curriculares correspondientes. En estas carreras es imprescindible la realización de transformaciones curriculares que se fortalezcan la formación en Matemática Aplicada de los futuros profesionales en cuestión; esto debido a que empresas que ofrecen productos o servicios, entidades gubernamentales y no gubernamentales demandan de la resolución de problemas de optimización de recursos o mejoramiento del servicio, así como de la modelización matemática.

Las universidades, y particularmente las Facultades y Departamentos docentes en cuestión, de forma periódica están pensando y materializando ciertos cambios, los cuales generalmente consisten en la incorporación y actualización de asignaturas o cursos como Modelos Diferenciales e Investigación de Operaciones. Con esto se trata de potencializar tanto el uso como el manejo de programas computacionales especializados, así como la integración de herramientas teóricas y prácticas que posibiliten la construcción pertinente de modelos matemáticos útiles para resolver problemas del ámbito productivo y servicios.

En este sentido los cambios más radicales en cuanto a la integración de cursos al plan de estudio, es el referido a Investigación de Operaciones. Como producto de estos cambios curriculares surge la problemática de la escasa existencia de materiales bibliográficos donde se aborden una forma intuitiva, sencilla y práctica los principales problemas de Programación Matemática y particularmente los problemas de Optimización Combinatoria.

Al revisar los libros de Investigación de Operaciones a los cuales tanto docentes como estudiantes tienen acceso, es posible evidenciar que la mayor parte de ellos abordan generalmente los problemas de Programación Lineal y Problemas de Redes, dejando a un lado los problemas de Optimización Combinatoria, que representan una gran parte de los problemas prácticos que se necesitan resolver y que cuya solución es demandada continuamente la sociedad, por las empresas que solicitan la resolución de problemas de optimización de recursos o mejoramiento del servicio.

Además de lo anterior se aprecia que los problemas son abordados con una metodología árida, descontextualizada, los problemas se resuelven de una solo forma o mediante un único algoritmo, y muy pocas veces o casi nunca se trabaja con algoritmos heurísticos, que son procedimientos que desempeñan un papel importante en la resolución de problemas de Optimización Combinatoria, con la idea de encontrar buenas soluciones factibles de manera rápida, cuando se tienen problemas cuyas instancias son de una gran dimensión y por tanto es el tipo de problema predominante la vida cotidiana.

Dada las condiciones antes mencionadas, surge la necesidad de responder a la siguiente interrogante ¿Cómo abordar de una forma intuitiva, sencilla y práctica los siguientes problemas clásicos de Optimización Combinatoria: Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Problema del Agente Viajero (Travelling Salesman Problem)?

1.3 Justificación

La gran cantidad de aplicaciones prácticas que tienen los problemas de Optimización Combinatoria hace que su tratamiento en los planes de estudios, programas de asignaturas, y sobre todos en las aulas de clases de las carreras de Ingeniería, Ciencias Exactas y Ciencas Económicas sea de fundamental importancia para responder en cierta medida a las continuas demandas que desde la sociedad llegan a los centros de estudios superiores.

En el aula de clases de las universidades, particularmente de las carreras antes señaladas existen muchos factores que deben de tomarse encuenta para que la generación de los aprendizajes sean robustos, significativos y desde luego que posibiliten la resolución de problemas en los distintos ámbitos laborales en donde deberán enfrentarse a situaciones donde deben optimizar problemas relacionados con la industria, economía, comercio, logística, informática, etc. Uno de estos factores es sin duda, la metodología utilizada para abordar los problemas que se consideran centrales en la asignaturas, en este caso Investigación de Operaciones.

En este sentido, la realización de este trabajo se justifica porque los resultados del mismo están encaminados a realizar un aporte a la solución del problema de la escasa o prácticamente nula existencia de libros u otros tipos de materiales donde se aborden los principales problemas de Optimización Combinatoria utilizando una metodolología intuitiva, sencilla y práctica para abordar los problemas en mención.

Con los resultados del trabajo se generan ideas que podrían tener un impacto directo en el proceso de mejoramiento de los programas de asignaturas del curso Investigación de Operaciones, el cual es ofertado en las Falcultades Ingeniería, Ciencias Econónicas y Ciencias Exactas de la distintas universidades del país. Mejoramiento, en el sentido de fortalecer la formación en Matemática Aplicada de los futuros profesionales de las Facultades antes mencionadas.

En lo que corresponde al aspecto disciplinar, la realización del trabajo se justifica por el hecho de facilitar el aprendizaje de los distintos algoritmos que se pueden tomar en cuenta para resolver algunos problemas clásicos de Optimización Combinatoria: el Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Eroblema del Agente Viajero (Travelling Salesman Problem).

1.4 Objetivos

1.4.1 Objetivo general

Proponer una metodología para el tratamiento de algunos problemas de Optimización Combinatoria: el Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Problema del Agente Viajero (Travelling Salesman Problem).

1.4.2 Objetivos específicos

- Revisar el estado del arte de la formulación matemática de ciertos problemas de Optimización Combintaria: Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de piezas y el Problema del Agente Viajero.
- Revisar el estado del arte de las técnicas de solución utilizadas algunos problemas de Optimización Combintaria: Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de piezas y el Problema del Agente Viajero.
- Implementar algoritmos exactos y heurísticos para resolver ciertos problemas de Optimización Combinatoria: Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de piezas y el Problema del Agente Viajero.
- Implementar el uso de herramientas computacionales (WinQsb) para resolver algunos problemas de Optimización Combinatoria: Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de piezas y el Problema del Agente Viajero.

CAPÍTULO 2

BREVE RESEÑA HISTÓRICA DE LA INVESTIGACIÓN DE OPERACIONES

2.1 Inicios de la Investigación de Operaciones antes del siglo XX

En el devenir de los tiempos el ser humano se ha visto constantemente en la necesidad de buscar la solución óptima para una variedad de problemas. En lo que respecta a procesos organizacionales, la necesidad de planificación y organización, en general, de optimización de recursos, aparece ya en el antiguo Egipto hacia el año 4000 A. C. y se va desarrollando a través de toda la antigüedad hasta el advenimiento del Imperio Romano. En China también aparecen tímidos movimientos de organización y dirección hacia el año 1000 A.C. Nabucodonosor establece algunas ideas sobre control de la producción hacia el año 600 A.C. Asímismo, en Grecia, se desarrollan en el años 350 A.C. los primeros métodos de organización del trabajo y del tiempo. Aproximadamente en el año 30 A.C., Julio César establece varias ideas de planificación, control y unidad de mando, que posteriormente implementa en todo el Imperio Romano. Todos los estudios y planteamientos organizacionales de la Antigüedad tienen su proyección, aunque no su continuación, a lo largo de toda la Edad Media.

En un contexto relacionado con Matemática, por ejemplo es posible mencionar que Euclides, en Los Elementos, expone las formas de determinar las líneas rectas de mayor y menor longitud, desde un punto hasta circunferencia de un círculo; además del paralelogramo de mayor área para un perímetro dado.

Figura 2.1: Braquistócrono

Entre el siglo XVII y XVIII, Joseph-Louis Lagrange plantea el problema de mostrar la existencia de una superficie mínima con una frontera dada; por su parte Jakob Bernoulli y Johann Bernoulli resuelven el problema de optimización de la curva de descenso más rápido o braquistócrona. Así mismo, Jean Baptiste-Joseph Fourier esbozó métodos de la actual programación lineal. Y en los últimos años del siglo XVIII, Gaspar Monge asentó los precedentes del método gráfico gracias a su desarrollo de la Geometría Descriptiva

Un hecho importante en la historia de la Investigación de Operaciones, que no es posible dejar de mencionar es la solución al problema de los puentes de Königsberg. Este famoso problema que reza de la siguiente manera: Dado el mapa de Königsberg, con el río Pregel dividiendo el plano en cuatro regiones distintas, que están unidas a través de los siete puentes, ¿es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes, recorriendo sólo una vez cada uno, y regresando al mismo punto de partida?

Leonhard Euler (1736) llega a la conclusión de que se trata de un problema irresoluble: no existe solución que permita hacer el recorrido pasando una sola vez por cada uno de los puentes. Estos estudios realizados por Euler fueron el detonante de la teoría de grafos, convirtiendo una simple discusión pueblerina en toda una disciplina científica. Euler demuestra una solución generalizada del problema, que puede aplicarse a cualquier territorio en que ciertos accesos estén restringidos a ciertas conexiones, tales como los puentes de Königsberg.

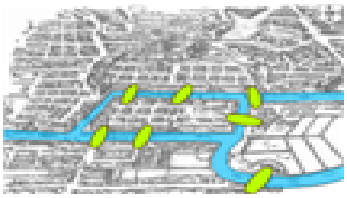


Figura 2.2: Puente de Königsberg

Es justamente, en los inicios de la primera Revolución Industrial, que el sentido y la forma de estudio de la Ciencia de la Gestión adquiere su ser más completo. Posteriormente, Adam Smith establece el principio de especialización en los trabajos, y Robert Owen, ya en el siglo XIX, realiza un estudio sobre tareas en un proceso productivo y advierte de la necesidad de adiestramiento en las mismas por parte de los operarios.

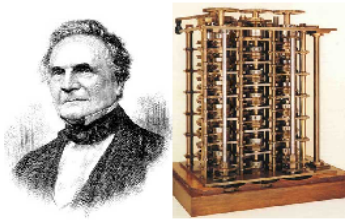


Figura 2.3: Babbage y su máquina diferencial

Una aportación fundamental la realiza Babbage, en 1832, construyendo lo que se podría llamar el primer computador digital, que vendría a ser el antecesor de los modernos ordenadores. También trabajó en la investigación de los costos de transporte y sistemas de clasificación del correo en England's Universal Penny Post en el año 1840.

A finales del siglo XIX, Joseph Wharton propone de la dirección estratégica e industrial un saber universitario. No obstante, el auge de las revoluciones industriales del siglo XIX permite asentar una base adecuada para el estudio de la ciencia operacional. Igualmente, Frederick W. Taylor y Henry L. Gantt, al experimentar la necesidad de planificación de la producción, establecen el método científico de dirección y las gráficas de programación productiva (de Gantt), respectivamente. A partir de este momento aparece la aportación nuclear del siglo XX a la Investigación de Operaciones, sabiendo que es en este siglo cuando ocurre su nacimiento real.

2.2 Comienzos de la Investigación de Operaciones en el siglo XX

En los inicios del siglo XX ocurrieron diversos acontecimientos que colaboraron con la génesis de la Investigación de Operaciones. Entre ellos es posible señalar a los siguientes hechos:

Figura 2.4: Agner Kraup Erlang

En 1909 se origina la teoría de colas, con Agner Kraup Erlang (Dinamarca, 1878 - 1929) en un esfuerzo para analizar la congestión de tráfico telefónico. El objetivo fundamental era cumplir la demanda incierta de servicios en el sistema telefónico de Copenhague; sus investigaciones acabaron en una nueva teoría denominada teoría de colas o de líneas de espera. Esta teoría es ahora una herramienta de valor en negocios debido a que un gran número de problemas pueden caracterizarse, como problemas de congestión llegada-salida.

John Von Neum(1903-1957), publica en 1928 una serie de artículos científicos que dieron origen a la Teoría de Juegos. Estos resultados fueron ampliados más tarde en su libro de 1944, The Theory of Games and Economic Behavior, escrito junto con Oskar Morgenstern. Dicha memoria contiene un método para encontrar soluciones óptimas para juegos de suma cero de dos personas. Durante este período, el trabajo sobre la temática en cuestión se centró, sobre todo, en teoría de juegos cooperativos. Este tipo de teoría analiza las estrategias óptimas para grupos de individuos, asumiendo que pueden establecer acuerdos entre sí acerca de las estrategias más apropiadas.

Figura 2.5: Johnvon Neumann

Figura 2.6: II Guerra Mundial-1944

Es importante indicar el momento en el cual se tuvo una conciencia clara de que algo nuevo y diferenciador estaba naciendo en el ámbito de la teoría de la organización en gran escala. Este momento es previo a la II Guerra Mundial. Podríamos decir que es hacia 1935 cuando Inglaterra se da cuenta de que necesita dar una respuesta adecuada al creciente poderío militar alemán. Por esta razón, el gobierno inglés urge un grupo de científicos a que realicen experimentos que conduzcan a un mejor

control del espacio aéreo. Fruto de esta experimentación aparece el radar, que constituye el inicio de la lucha por la supremacía aérea. Este grupo de investigadores tomó su base en Bawdsey y por esta razón se llamó grupo de Bawdsey.

De forma paralela, otro grupo se estuvo estableciendo durante 1936 para desarrollar el experimento Biggin Hill, que permitía la simulación de aviones enemigos y su detección. La conjunción de estos dos grupos, permitió ofrecer a la RAF (Royal Air Forcé) una estructura operacional para sus equipos materiales y humanos, que le posibilitó librar la batalla de Inglaterra en $1940 - 1941$. El grupo de Bawdsey fue dirigido en 1938 por A.P. Rowe, el cual acuñó la expresión "Operations Research", que posteriormente se extendió dentro del ámbito científico al resto de países occidentales.

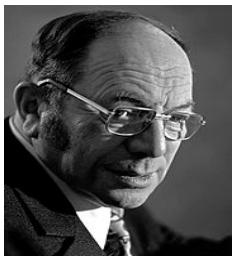


Figura 2.7: Leonid Kantorovich 1912-1986

En 1939, el matemático ruso Leonid V. Kantarovich publica sus trabajos bajo el título "Métodos matemáticos de planificación y organización de la producción", el cual se exponen métodos matemáticos de la programación lineal, aplicable para maximizar la eficacia de variables económicas tales como la productividad, las materias primas y el trabajo. Este trabajo no llegó a ser conocido en occidente hasta bastantes años después y de hecho no fue traducido al inglés hasta 1959. Casi de forma simultánea, el matemático holandés Koopmans, se plantea un problema similar y obtiene resultados parecidos. Por ello el "problema del transporte" es conocido hoy como "problema de Koopmans-Kantarovich". Es importante señalar posteriormente (1975) seria galardonado con el Premio Noble de Economía en junto a Koopmans por sus teorías sobre la asignación óptima de recursos escasos. Sus teorías fueron utilizadas para mejorar la planificación económica y la distribución de recursos en la Unión Soviética.

La batalla de Inglaterra se recrudece en el otoño de 1940. La Luftwaffe, Fuerza Aérea Alemana, estaba sometiendo a este país a un fuerte acoso aprovechando la reducida capacidad aérea británica debido a la política de desarme, aunque experimentada en el combate. El gobierno británico, buscando algún método para defender su país, convocó científicos de diversas disciplinas para tratar de resolver el problema y sacar el máximo beneficio de los radares de reciente invención de que disponían. El físico, P.M.S. Blackett de la Universidad de Manchester fue responsabilizado de formar un grupo de trabajo para estudiar el sistema de defensa antiaérea gobernado por radar. Este grupo, estaba constituido por tres psicólogos, dos físicos matemáticos, un astrofísico, un oficial del ejército, un topógrafo, un físico y dos matemáticos. Gracias a su trabajo determinando la localización óptima de las antenas y la mejor distribución de las señales consiguieron duplicar la efectividad del sistema de defensa aérea y evitar que la isla cayera en manos de la Alemania nazi.

Figura 2.8: Bombarderos alemanes

Cuando los Estados Unidos entran en la guerra, son conscientes de la necesidad de tales grupos operativos y de la constitución de secciones operacionales para el éxito de los mismos. De esta manera, constituyen en 1942 un grupo operacional de lucha antisubmarina (ASWORG - Anti-Submarine Warfare Operations Research Group) que recoge toda la experiencia inglesa desarrollada por Blackett.

También en 1942, la U-Bootswaffe alemana con su flota de submarinos U-Boot inició un bloqueo a Gran Bretaña atacando convoyes de barcos cargados de suministros procedentes de Estados Unidos e impidiendo que alcanzaran su destino. Para tratar de solventar la situación, la ASWORG realizó representaciones matemáticas de dichos convoyes, teniendo en cuenta una serie de restricciones y condiciones impuestas por la realidad, tales como la velocidad máxima a la que podían desplazarse los navíos, la cantidad de suministros que debían transportar, y el combustible necesario para alcanzar su destino.

Figura 2.9: U-Boot en combate

Aplicaron estos modelos también sobre los U-Boots: el tamaño de su flota, el alcance de los submarinos, sus torpedos, etc. Con base a esta información fueron capaces de modelar la guerra naval, y determinar si era mejor una estrategia basada en convoyes formados por un gran grupo de navíos de carga escoltados por muchos destructores, o por el contrario pequeños grupos más difíciles de localizar para el enemigo, e incluso la manera de causar un mayor

daño a los submarinos U-Boot. Cuando la armada de los Estados Unidos de América puso en práctica esta estrategia, disminuyó de forma considerable la cantidad de barcos hundidos mientras se incrementaba la destrucción de submarinos alemanes (pasando del hundimiento de apenas una treintena al año a rondar los 250 anuales en 1943 y 1944).

No puede decirse que las potencias del Eje (Alemania, Japón, Italia) hicieran uso de las técnicas operacionales durante la II Guerra Mundial, mientras que el número de científicos e investigadores involucrados en Investigación de Operaciones en la contienda por parte de ingleses, americanos y canadienses superó los setecientos. Las aportaciones que hicieron todos estos investigadores supusieron un cambio radical en la forma de percibir la Ciencia de la Gestión en los años siguientes.

De alguna manera, todos estos estudiosos que trabajaban de manera aislada en los años treinta, se aglutinaron holísticamente con ocasión de la guerra, y produjeron un conjunto de técnicas y teorías que ocasionaron el alumbramiento de la Investigación de Operaciones como ciencia.

2.3 Desarrollo de la Investigación de Operaciones de 1945 a la actualidad

Es tarea complicada tratar de resumir en algunos párrafos todo lo que han supuesto las décadas anteriormente mencionadas para la Investigación de Operaciones. Realmente, se ha construido más ciencia operacional durante estos años que en todo el resto de la historia de la humanidad. Puede decirse, por tanto, que la verdadera historia de la Investigación de Operaciones se ha desarrollado durante este período: se han establecido líneas de investigación, han aparecido sociedades profesionales, se han creado revistas de investigación, se han publicado libros y se ha incluido la materia dentro del curriculum educativo.

Una vez finalizada la guerra mundial y puesto en evidencia el éxito cosechado por las técnicas operacionales, éstas continuaron desarrollándose dentro del ámbito militar, puesto que era el ejército quien poseía la mayor parte de los investigadores, y quien estaba interesado en proseguir dicha línea de trabajo. Como muestra de dicho desarrollo, a continuación se continúan mencionando hechos relevantes en la historia de la Investigación de Operaciones.

Figura 2.10: George Stigler 1911-1991

En 1945, George Joseph Stigler planteó el problema de la dieta, a raíz de la preocupación del ejército americano por asegurar unos requerimientos nutricionales básicos para sus tropas al menor coste posible. Se trataba de determinar la cantidad, entre 77 alimentos diferentes, que debería ingerir diariamente un hombre mediano de aproximadamente $70Kg$ de peso, de modo que las necesidades mínimas de nutrientes fuesen iguales a las recomendadas por el Consejo Nacional de Investigación norteamericano. El problema fue resuelto manualmente mediante un método heurístico con el cual se examinaron 510 diferentes posibilidades de combinación de alimentos, y cuya solución difería tan sólo unos céntimos de la solución aportada años más tarde por el método Simplex.

En 1947, los Estados Unidos ponen en marcha el Proyecto Scoop (Scientific Computation Of Optima Programs) el cual surgió como medida bélica en la Segunda Guerra Mundial, adelantándose a la Unión Soviética, por parte de los norteamericanos, y más concretamente por un colectivo de científicos dirigidos por George Dantzig, en 1947. Dicho proyecto lo encargó la Fuerza Aérea y trata de resolver problemas de Programación Lineal, y como producto de éste, vio la luz el Método Simplex, que es el método más conocido de resolución de problemas de Programación Lineal, y que es aún muy utilizado. Actualmente, debido al crecimiento de la capacidad de cálculo de los computadores, ha permitido el uso de las técnicas desarrolladas en problemas de gran dimensión.

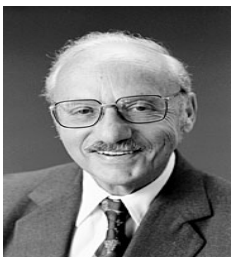


Figura 2.11: Dantzig 1914-2005

Figura 2.12: MIT 1948

A mediados de los años cincuenta se desplazó el centro de gravedad de interés de la Investigación de Operaciones, y alcanzó el terreno industrial y el académico. En lo que respecta a este último ámbito, los primeros cursos sobre Investigación de Operaciones se impartieron en el M.T.I. (Massachusetts Institute of Tecnology) de Boston en 1948. Y solo un año después 1949, hubo un ciclo de conferencias en el University College de Londres. Poco después, ofrecían programas específicos completos las Universidades Case Western Reserve, John Hopkins y North-Western en U.S.A.; y en el Imperial

College y la London School of Economics en Inglaterra.

Al mismo tiempo que la disciplina de la Investigación de Operaciones, se desarrollaron también las técnicas de computación, las cuales permitieron una importante reducción del tiempo de resolución de los problemas. El primer resultado de estas técnicas se obtuvo en el año 1952, utilizando un ordenador SEAC (Standards Eastern Automatic Computer) del Instituto Nacional de Estándares y Tecnología de los Estados Unidos, para obtener la solución de un problema. El éxito en el tiempo de resolución fue tan alentador que de inmediato se usó para todo tipo de problemas militares tales como la gestión de fondos monetarios para logística y armamento, determinar la altura óptima a la que deberían volar los aviones para localizar los submarinos enemigos, e incluso la profundidad a la que se debían enviar las cargas para alcanzar los submarinos enemigos de forma que causara el mayor número de bajas. Todo esto se tradujo en un aumento de hasta cinco veces en la eficacia de la fuerza aérea.

A partir de 1950, un número cada vez mayor de investigadores (matemáticos y economistas) aislados o constituyendo grupos contribuyen al desarrollo de las diferentes ramificaciones de la Programación Lineal; en particular, la "Rand Corporation" con G. B. Dantzig y W. Orchard-Hays, después L. R. Ford, D. R. Fulkerson, y D. Gale; el Departamento de Matemáticas de la Universidad de Princenton con A. W. Tucker y H. W. Kun; la "Graduate School of Industrial Administration" del "Carnegie Institute of Technology" con A. Charnes y W. Cooper.

Los dos primeros grupos trabajan en la teoría matemática de los programas y su instalación en computadoras; los resultados se publicaron en la "Rand Corporation" en la serie de "Rand notes on linear programming and extensions". De estas publicaciones se deben mencionar las de Dantzig sobre los desarrollos teóricos, las de W. Orchard Hays sobre la instalación de los programas de cálculo en máquinas, las de L. R Ford y D. R. Fulkerson sobre las redes de transporte; es necesario citar especialmente en el activo del grupo de Princenton, el método "húngaro" de H. W. Kun, para los problemas de asignación, la publicación de la notable colección de notas "Linear Inequalities and Related Systems" en 1956 y el método de Gomory para el cálculo de los problemas lineales en números enteros a finales del año 1958.

El equipo del "Carnegie Tech" desarrolló la Programación Lineal en aplicaciones industriales, se interesó en aspectos teóricos particulares como: degeneración, errores de redondeo, el Simplex revisado, variables acotadas.

En 1953, Richard Bellman publica su primer libro en relación con la programación dinámica, "*An introduction to the theory of dynamic programming*". La invención de este tipo de progrmación fue un avance muy importante que sentó las bases para la aplicación de técnicas de ecuaciones funcionales en áreas más allá de las que imaginó Bellman.Los aportes de este matemático a la Programación Dinámica marcaron el inicio de una nueva era en análisis y optimización de sistemas a larga escala y abrió el paso para la aplicación de sofisticadas técnicas computacionales en problemas como el diseño de sistemas guía para vehículos espaciales, el control de plagas y la optimización de la red. La ecuación de Bellman se aplicó para la teoría de la ingeniería de control, temas de matemáticas aplicadas y posteriormente se convirtió en una herramienta de suma importancia para la teoría económica.

Figura 2.13: Richard B.1920-1984

Figura 2.14: Frank Rosenblatt 1928-1969

En 1958, Frank Rosenblatt, psicólogo estadounidense, plantea las bases de Las Redes Neuronales con su publicación "*The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*", en la Psychological Review, de Estados Unidos. Esta experiencia, alentó la creencia de que no estaba lejano el desarrollo de máquinas inteligentes, en las que se reprodujesen de forma compleja las funciones observadas por la neurobiología.

En la década de los setenta continuó el desarrollo expansivo de la Investigación de Operaciones, llegando al ámbito de la administración pública, tratando los siguientes tipos de problemas: transporte urbano, administración de justicia, construcción de edificios públicos, educación, hospitales y servicios sociales. En lo que respecta al desarrollo como ciencia, es importante señalar que en 1975, el matemático norteamericano John Henry Holland publica "*Adaptation in Natural and Artificial System*" , trabajo del cual surgió una de las líneas más prometedoras de la inteligencia artificial, la de los algoritmos genéticos. Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones, y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados. Los algoritmos genéticos se enmarcan dentro de los

Figura 2.15: John Henry Holland 1929-2015

algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.

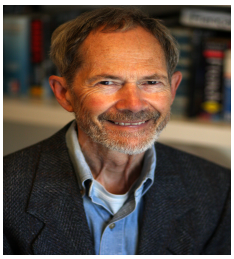


Figura 2.16: Fred Glover 1937

Otro hecho relevante en los setenta, fue la publicación en 1977 por Fred Glover y Manuel Laguna de "*Tabu Search*". Este procedimiento surge, en un intento de dotar de "inteligencia" a los algoritmos de búsqueda local. Según Fred Glover, su primer definidor, "la búsqueda tabú guía un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local". Esta técnica toma de la Inteligencia Artificial el concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta, es decir, el procedimiento trata de extraer información de lo sucedido y actuar en consecuencia. En este sentido puede decirse que hay un cierto aprendizaje y que la búsqueda es inteligente.

Dos años después de la publicación del Algoritmo Búsqueda Tabú, en 1979, Khachiyan publica "*A polynomial algorithm in linear programming*", llamado Algoritmo elipsoidal. Este resultado causó gran entusiasmo en el mundo académico dedicado a estas temáticas, puesto que implicaba la solución en tiempo polinomial de los problemas de programación lineal. Aunque este algoritmo actualmente se considera poco práctico, por el alto grado polinómico de su complejidad computacional, significó un descubrimiento sumamente importante en el área, y un punto de partida para el desarrollo de algoritmos más sofisticados y eficientes.

Figura 2.17: Leonid Khachiyan 1952-2005

En la década de los ochenta, de forma independiente Kirkpatrick, Gelatt y Vecchi (1983) y Cerny (1985), publican sus trabajos referidos al método Simulated annealing (recocido simulado), el cual es un algoritmo de búsqueda meta-heurística para problemas de optimización global; el objetivo general de este tipo de algoritmos es encontrar una buena aproximación al valor óptimo de una función en un espacio de búsqueda grande. A este valor óptimo se lo denomina óptimo global. El nombre e inspiración viene del proceso de recocido del acero y cerámicas, una técnica que consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas.

El calor causa que los átomos aumenten su energía y que puedan así desplazarse de sus posiciones

iniciales (un mínimo local de energía); el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la inicial (mínimo global).

Figura 2.18: Narendra Karmarkar 1957

En 1984, Narendra Karmarkar publica el artículo "*A New Polynomial-Time Algorithm for Linear Programming*" , el cual representó todo un hito en los procesos de optimización lineal. A partir del trabajo de Karmarkar se reactivó la investigación en optimización lineal, especialmente en métodos de barrera y de punto interior, también para optimización no lineal. El algoritmo de Karmarkar se presenta como un buscador de óptimos a partir de puntos interiores, siendo ésta la gran novedad en relación con el método simplex. En la publicación no se describe totalmente el método resolutorio y, además, afirma que es mucho más rápido que el simplex para problemas de gran dimensión.

2.3.1 Grupos científicos sobre Investigación de Operaciones

Una demostración de la maduración de una ciencia lo constituye la formación de estables organizaciones que agrupen a los más prestigiosos profesionales de la materia. Así, en Inglaterra, en abril de 1948, se funda el Operational Research Club, como un lugar para discutir y conversar acerca de cuestiones operacionales por parte de los científicos que habían tomado parte en la segunda guerra mundial. El propósito central del grupo era proporcionar una estructura organizativa en la que tuvieran lugar seis reuniones anuales en la sede de la Royal Society de Londres.

En Estados Unidos a principios de los años cincuenta se crea ORSA (Operations Research Society of America, 1952) y TIMS (The Institute of Management Science, 1953), las cuales se fusionan en 1995 para establecer INFROMS. El nacimiento de ORSA tiene lugar en una reunión en mayo de 1952 en Arden House, Harriman, New York. El académico Philip M. Morse, fue elegido el primer presidente. Dicha sociedad publicó la revista Journal of Operations Research Society of America, que en 1956 pasó a llamarse simplemente Operations Research.

En 1953, la labor de ORSA se vio completada con la de TIMS, sociedad de carácter internacional y cuyo primer presidente fue William W. Cooper. Dicha sociedad publicó el primer número de la revista Management Science en septiembre de 1954, siendo C. West Churchman su primer editor. El desarrollo y la colaboración de estas dos sociedades propició que en septiembre de 1957 se celebrara en la Universidad de Oxford, una de las primeras reuniones de carácter internacional de Investigación

de Operaciones con 250 delegados de 21 países.

Fruto de las primeras reuniones internacionales, en 1959 se constituye IFORS (International Federation of Operational Research Societies), federación de las sociedades ORSA, Operations Research Society y Société Frangaise de Recherche Opérationelle. En la actualidad, más de treinta y cuatro sociedades nacionales e internacionales se encuentran vinculadas a IFORS.

Una relación detallada de algunas de sociedades operativas existentes en el mundo y del año de su fundación se da a continuación en la tabla siguiente:

Año	**Sociedades Nacionales**	**Año**	**Sociedades Nacionales**	**año**	**Sociedades Nacionales**
1959	Francia, Reino Unido, Estados Unidos	1975	Chile,Finlandia	1992	Hungría
1960	Australia, Bélgica Canadá. India, Países Bajos.	1976	Egipto	1993	Bulgaria
1961	Suecia, Noruega	1977	Turquía	1994	Croacia, República Chequa Eslovaquia
1962	Argentina, Alemania, Italia	1978	Singapur	1998	Belarús
1963	Dinamarca, España, Suiza	1979	Austria	2002	Bangladesh Colombia, Lituania
1966	Grecia, Irlanda, México	1982	China, Portugal	2007	Eslovenia
1969	Brasil, Israel	1983	Hong Kong, Yugoslavia	2009	Uruguay, Irán
1970	Nueva Zelanda	1986	Islandia	2010	Estonia, Perú
1972	Corea	1988	Malasia	2011	Nepal, Túnez
1973	Sudáfrica	1990	Filipinas, Polonia	2015	Nigeria

Tabla 2.1: *Afiliación a IFORS*

La idea de reuniones más frecuentes entre investigadores de diferentes regiones del planeta generó que se constituyeran agrupaciones regionales, como las siguientes:

- Asociación de Sociedades Europeas de Investigación Operativa dentro de IFORS (EURO). Constituida en Bruselas en 1975.
- Asociación Latinoamericana de Sociedades de Investigación Operativa (ALIO). En 1982 Establecida en 1982, en Río de Janeiro, después de que los trabajadores de Argentina, Brasil y Chile sentaran las bases de un grupo regional el año anterior. ALIO también cuenta entre sus miembros dos sociedades nacionales que son simultáneamente miembros de EURO, a saber, la sociedad nacional de España (SEIO) y la de Portugal (APDIO) (Yanasse, 2008).
- Asociación de las entidades del Pacífico Asiático en IFORS (APORS) fundada en 1985, y está formada por representantes de Corea, China, Japón y Australia.
- Asociación de Sociedades Nacionales de Norteamérica (NORAM). Se fundó en 1987 y está compuesta por las Sociedades de OR en Canadá (CORS) y los EE.UU. (INFORMS).

Un aspecto importante por mencionar, es que como producto de las interacciones entre los académicos pertenecientes a las diversas sociedades antes mencionadas, el termino Investigación de Operaciones (IO) se utiliza de forma equivalente al de Programación Matemática (PM), Investigación Operativa, Gerencia de Operaciones (GP). Los fines de la disciplina son los mismos, la forma de nombrarle depende del país o la región. Es por ello que de aquí en adelante, en el documento, se usará indistintamente cualquiera de los términos.

Esta abundancia de sociedades demuestra el dinamismo de la Investigación de Operaciones, puesto que permite decir que desde la década de los setenta se ha constituido en una ciencia extendida universalmente. La profusión de revistas y libros ha contribuido a ello sobremanera, puesto que ha sido el nexo de unión y comunicación entre estudiosos e investigadores. Este será el aspecto que a continuación vamos a tratar.

2.3.2 Revistas y libros en la divulgación de la Investigación de Operaciones

Una de las primeras revistas publicadas en el campo de la Investigación de Operaciones fue Operational Research Quarterly, editada por la Operational Research Society a partir de 1950. La siguiente fixe Operatians Research, publicada en América por ORSA. A partir de 1955 ya son numerosas las revistas de Investigación de Operaciones que aparecen en los países occidentales. No obstante, a

partir de 1965 el número de artículos operacionales publicados crece de manera exponencial en el mundo científico.

Por su gran utilidad como revista recopiladora de artículos de Investigación de Operaciones se encuentra International Abstraéis in Opérations Research (IAOR) creada en 1961, que publica resúmenes de notas y artículos de Investigación de Operaciones de todas las revistas del mundo. Aunque inicialmente IAOR es publicada por ORSA, en la actualidad realiza esta labor North-Holland Publishing Company (Amsterdam) para IFORS. Esta labor de recopilación para los años anteriores a 1961 fue realizada por James H. Batchelor, dando lugar a la publicación de una obra en tres volúmenes. De esta forma LAOR, junto con los estudios de Batchelor, permite disponer de una relación actualizada de toda la literatura de Investigación de Operaciones en el mundo.

De igual modo, las revistas más importantes de Investigación de Operaciones en el mundo, de acuerdo a su interés histórico son las siguientes (entre paréntesis se ha señalado el año de su fundación):

- Operational Research Quarterly (1950)
- Opérations Research (1952)
- Naval Research Logistics Quarterly (1954) Management Science (1954)
- Revue Française d Automatique, Informatique, Recherche Operationelle (1956)
- Zeitschrifí fiir Opérations Research (1956)
- Journal of the Opérations Research Society of Japan (1957)
- Interfaces (1971)
- INFOR (Journal of the Canadian Operational Research Society) (1971)

Actualmente, INFORMS publica doce revistas académicas sobre Investigación de Operaciones, incluyendo las dos mejores revistas de su clase, de acuerdo con Journal Citation Reports (2005). Estas son

- Análisis de Decisiones
- Information Systems Research
- INFORMS Diario de Computación

- Interfaces
- Gestión de la Ciencia
- Manufactura y Servicio de Gestión de Operaciones
- Marketing Science
- Matemáticas de Investigación de Operaciones
- La Investigación De Operaciones
- Organización de Ciencias
- Transporte Ciencia y
- INFORMA Transacciones en Educación: revista de acceso abierto.

Otras revistas que presentan resultados importantes sobe IO son:

- European Journal of Operational Research (EJOR): Fundada en 1975 y actualmente es de lejos la más grande revista Investigación de Operaciones en el mundo con su alrededor de 9.000 páginas de documentos publicados por año. En 2004 , el número total de cita fue el segundo más grande entre las revistas Investigación y ciencia gestión operativa.
- Diario de la Sociedad de Investigación Operativa (jors): es una publicación oficial de la Sociedad OR.
- INFOR Diario: publicado y patrocinado por la Sociedad de Investigación de Canadá Operacional.
- Opsearch: revista oficial de la Sociedad de Investigación Operativa de la India.
- TOP: Diario Oficial de la Sociedad Española de Estadística e Investigación Operativa.
- JDMS: El Diario de Defensa de Modelado y Simulación: Aplicaciones, Metodología, Tecnología. Revista trimestral dedicada al avance de la ciencia de la modelización y la simulación en su relación con los militares y de defensa.

En lo referente a libros, existe actualmente una gran cantidad de obras que abordan los problemas Investigación de Operaciones. Aquí únicamente se pretende mostrar aquellos textos que supusieron un aporte básico desde el punto de teórico e histórico. Una aportación fundamental dentro del crecimiento y expansión de la IO, fue la publicación de los conocimientos operacionales conseguidos

en la II Guerra Mundial, a través de diversas obras como: Methods of Operations Research (1946) de Morse y Kimball, Profile in Operations Research (1973) de Johnson y Operational Research in War and Peace (1973) de Katcher y Waddinton. La primera de ellas fue durante veinticinco años una obra de obligada referencia para todos los profesionales de la gestión operativa. No cabe la menor duda, que es necesaria la sedimentación de conocimientos antes de escribir unos manuales que expongan los hitos y pautas fundamentales de la Investigación de Operaciones. Por esta razón, la elaboración de libros es posterior en el tiempo a la creación de revistas. Una pléyade de libros con grandes aportaciones en el campo de la IO ve la luz a finales de los años cincuenta.

Por destacar los más importantes podemos citar:

- Koopmans (1951) "Activity Analysis of Production and Allocation"
- McKinsey (1952) "Introduction to the Theory of Games"
- Whitin (1953) "The Theory of Inventory Management"
- Manne (1956) "Scheduling of Petroleum Refinery Operations"
- Bellman (1957) "Dynamic Programming"
- Dorfman, Samuelson y Solow (1958) "Linear Programming and Economic Analysis"
- Saaty (1959) "Mathematical Methods of Operations Research"

Por supuesto, a partir de los años sesenta, la Investigación de Operaciones ya se encontraba totalmente separada del ámbito militar y constituye claramente una ciencia civil. El desarrollo e historia de la IO está vinculada de manera estrecha a cada una de sus especialidades. Por ello, resulta interesante conocer una división clásica de las ramas relacionadas con la Investigación Operativa de acuerdo con el tipo de modelos que manejan:

Modelos Determinísticos	**Modelos estocásticos**
• Programación Lineal • Programación Entera. • Teoría de Grafos. • Flujos en Redes. • Programación Geométrica. • Programación No Lineal. • Programación a Gran Escala. • Teoría de Control Óptimo.	• Procesos Estocásticos. • Teoría de Colas. • Teoría del Valor • Análisis de Decisión. • Teoría de Juegos. • Teoría de Búsqueda. • Simulación. • Programación Dinámica. • Teoría de inventarios

Tabla 2.2: *Modelos de la Investigación de Operaciones*

2.3.3 La práctica de la Investigación de Operaciones

Las cuestiones prácticas fueron las que contribuyeron al nacimiento de la Investigación de Operaciones. Sin ellas, la ciencia operacional hubiera tenido otro elemento constitutivo. Aunque no hay que despreciar los grandes desarrollos teóricos de muchas de las áreas de la Investigación Operativa, sin los cuales algunos resultados no se hubieran dado; es necesario contemplar la Investigación de Operaciones como una ciencia fundamentalmente práctica. Desafíos prácticos han inspirado multitud de avances metodológicos y teóricos.

Por ejemplo, hace años las necesidades de distribución de gas natural llevaron al desarrollo del método de descomposición de Dantzig-Wolfe para programas no lineales. Muchos otros problemas prácticos como los de distribución de energía, mercados monetarios, mercados agrícolas, equilibrio del tráfico urbano, dieron lugar a otros tipos de modelos con sistemas de desigualdades. En la cara opuesta de la moneda, las nuevas aportaciones en Programación Entera han permitido la resolución de un amplio rango de problemas.

Afortunadamente, para conocer los principales trabajos prácticos en Investigación de Operaciones se puede consultar la revista Interfaces que publica bimestralmente INFORMS. Esta revista presenta cada año en su primer número, el trabajo ganador del premio Edelman (en honor al científico alemán Franz Edelman) a la mejor aplicación de la Investigación de Operaciones en cualquier campo técnico: industria, gestión pública, decisiones estratégicas, etc. Como ilustración de lo anteriormente expuesto acompañaremos estos comentarios con algunas ideas sobre dos trabajos que han ganado este premio:

- Corte de árboles de manera óptima en Weyerhauser: En una tala de árboles de tipo industrial, una compañía maderera tiene que tomar decisiones sobre cómo cortar árboles talados en troncos sin ramas con objeto de transportarlos a plantas de procesamiento que los transformen en productos finales. Estas decisiones son importantes, puesto que el valor de un árbol varía hasta en un cincuenta por ciento o más dependiendo de cómo esté cortado. La decisión de cortado es complicada, puesto que depende de las características de los árboles talados (longitud, curvatura, diámetro...) así como del valor final del producto acabado. Los analistas de Weyerhauser Company formularon este problema mediante un programa dinámico y diseñaron un juego con video interactivo que permitía jugar contra el programa dinámico con objeto de probar y refinar las pruebas heurísticas elegidas. Esta técnica ha contribuido a un aumento de beneficios de más de 100 millones de dólares en diez años.

- Gestión de los recursos acuíferos de Holanda: La gestión de agua a nivel nacional es un problema que debe buscar un equilibrio entre diversas consideraciones económicas, políticas y técnicas, por un lado, y la incertidumbre en la disponibilidad y consumo de agua. El problema es particularmente importante para un país como Holanda, que depende fuertemente de la producción de cosechas de regadío para su bienestar económico. Con objeto de resolver este problema se han planteado seis programas matemáticos y dos procedimientos de optimización heurística. Este planteamiento integrado permitió el ahorro de miles de millones de dólares en gastos de inversión, y redujo el costo del daño a la agricultura en cerca de 15 millones de dólares.

Muchos otros trabajos han aparecido bajo el amparo de INFORMS, sin embargo, los premios otorgados a importantes trabajos de Investigación Operaciones, bien sea por su interés aplicado, bien por el desarrollo de nuevas teorías, han supuesto un importante estímulo a la hora de planificar y dirigir líneas concretas de investigación. Algunos de estos premios han tenido un importante papel histórico y han estado más de moda en unos años que en otros. Podemos citar, entre otros, los premios siguientes:

- Premio Lanchester, otorgado anualmente por ORSA (ahora que se ha integrado en INFORMS, lo concede dicho instituto) por la mejor publicación en Investigación de Operaciones en lengua

Inglesa. Se concede desde 1954, y en 1994 tenía una dotación monetaria de 5000.

- Premio Dantzig, concedido cada tres años por la Mathematical Programming Society y la Society for Industrial and Applied Mathematics por la mejor contribución original al campo de la Programación Matemática.
- Premio Fulkerson, otorgado cada tres años por la American Mathematical Society y la Mathematical Programming Society para el artículo de investigación más brillante en el campo de las Matemáticas Discretas.
- Premio Orchard-Hays, concedido cada tres años por la Mathematical Programming Society por la excelencia en Programación Matemática Computational.
- Premio Franz Edelman, otorgado por TIMS (ahora por INFORMS) a los mejores trabajos de puesta en práctica de la Investigación Operacional. Los galardonados con el primer premio tienen una dotación económica de 10000 y su aplicación es publicada en la revista Interfaces.

La Investigación de Operaciones forma cada día más, una parte de las actividades normales de la empresa moderna y, por tanto, ya no se trata de una función especializada que deba llevarse a cabo en un departamento separado. En la siguiente tabla se pueden observar algunos ejemplos de casos reales de uso de la Investigación de Operaciones por parte de diferentes organizaciones y las ganancias y/o ahorros conseguidos a raíz de ello.

Organización	**Aplicación**	**Año**	**Ahorros anuales**
UPS	Proyecto de optimización y navegación integrada en carretera Onion (ORION)	2016	$300 a $400 millones anualmente.
Syngenta	Buen crecimiento a través del análisis avanzado	2015	Aumento en la tasa de producción
U.S. Centers for Disease Control and Prevention	Erradicadores de la polio utilizan modelos analíticos integrados para tomar mejores decisiones	2014	40 millones de dólares en los países cubiertos por la GPEI

Tabla 2.3: Fuente: INFORMS

Organización	Aplicación	Año	Ahorros anuales
Operador Independiente del Sistema de Transmisión de Midwest (MISO)	Desbloquea miles de millones en ahorros a través de la aplicación de la investigación de operaciones para mercados de servicios energéticos y auxiliares	2011	$ 3.0 mil millones en ahorros acumulados de 2007 a 2010.
INDEVAL	Desarrolla un nuevo sistema operativo y de liquidación mediante la investigación de operaciones	2010	Mejora y fortalece la infraestructura financiera de México.
Hewlett-Packard HP	transforma la gestión de la cartera de productos con la investigación de operaciones	2009	$500 millones de ganancias adicionales
Netherlands Railways	El nuevo horario holandés: The O.R. Revolución	2008	Ganancia anual adicional de 40 millones de euros
Memorial Sloan-Kettering Cancer Center (MSKCC)	Investigación de operaciones avanza en terapias contra el cáncer	2007	El tiempo de la sala de operaciones se acorta, y todo el procedimiento es menos invasivo.
Warner Robins Air Logistics Center	El Centro de Logística Aérea de Warner Robins simplifica la reparación y revisión de aeronaves	2006	Ahorra a los contribuyentes estadounidenses $ 50 millones al año.
General Motors	General Motors incrementa su producción	2005	$ 2.1 mil millones en más de 30 plantas de vehículos y 10 países.

Tabla 2.4: Fuente: INFORMS

Organización	Aplicación	Año	Ahorros anuales
Motorola Inc.	Reinventar el Proceso de Negociación de Proveedores en Motorola	2004	$ 600 millones de dólares hasta la fecha
Continental Airlines	Optimización de la reasignación de tripulantes cuando ocurren interrupciones en el itinerario.	2003	$40 millones
Samsung Electronics	Desarrollo de métodos para reducir los tiempos de manufactura y niveles de inventario.	2002	$200 millones de ganancias adicionales
Merrill Lynch	Diseño de los precios de opciones basadas en activos y en línea directa para proporcionar servicios financieros.	2001	$80 millones de ganancias adicionales.
IBM	Reingeniería de su cadena global de suministros para responder con mayor rapidez a los clientes mientras se mantiene el mínimo de inventarios.	2000	$750 millones en el primer año
Sear, Roebuck	Desarrollo del sistema de rutas e itinerario para el servicio de transporte y entrega a domicilio.	1999	$42 millones
Hewlett-Packard	Rediseño de tamaño y localización de inventarios de seguridad en la línea de producción de impresoras para cumplir metas de producción.	1998	$280 millones de ingreso adicional

Tabla 2.5: Fuente: INFORMS

Organización	Aplicación	Año	Ahorros anuales
Taco Bell	Programación óptima de empleados para proporcionar el servicio a cliente deseado con un costo mínimo.	1998	$13 millones
Procter & Gamble	Rediseño del sistema de producción y distribución norteamericano para reducir costos y mejorar la rapidez de llegada al mercado.	1997	$200 millones
Cuerpo de defensa de la República de Sudáfrica	Rediseño óptimo del tamaño y forma del cuerpo de defensa y su sistema de armas.	1997	$1.100 millones
China	Selección y programación óptima de proyectos masivos para cumplir con las necesidades futuras de energía del país.	1995	$425 millones
Digital Equipment Corp.	Reestructuración de toda la cadena de suministros entre proveedores, plantas, centros de distribución, sitios potenciales y áreas de mercado.	1995	$800 millones
Delta Airlines	Maximización de ganancias a partir de la asignación de los tipos de aviones en 2.500 vuelos nacionales en Estados Unidos.	1994	$100 millones

Tabla 2.6: Fuente: INFORMS

Toda la información mostrada en la tabla anterior vienen a poner en evidencia que la Investigación Operativa está íntimamente relacionada con características tales como funcionamiento de sistemas, toma de decisiones, búsqueda de solución óptima, equipos interdisciplinarios, modelización matemática y desarrollo computacional. Esta caracterización se ve reflejada en la conceptualización de Investigación de Operaciones, que a lo largo de décadas, han presentado diversos autores. A continuación se presentamos algunas de estas definiciones.

2.4 ¿Qué es la Investigación de Operaciones?

Kaufmann (1965), en su libro Métodos y Modelos de la Investigación de Operaciones, considera que la Investigación de Operaciones es la ciencia de la preparación de decisiones. En esta conceptualización claramente se aprecia que el peso que le da el autor a la toma de decisiones.

Asimismo Ackoff (1971) afirma que la Investigación Operativa puede caracterizarse como la aplicación de métodos, técnicas e instrumentos científicos a problemas que implican el funcionamiento de sistemas, para proporcionar soluciones óptimas a las personas que controlan las operaciones. En este mismo año, Sasieni señala que la Investigación de Operaciones se puede considerar como la aplicación del método científico por equipos interdisciplinarios a problemas que comprenden el control de sistemas organizados hombre-máquina, para dar soluciones que sirvan mejor a los propósitos de la organización como un todo. Las características esenciales señaladas en estas dos conceptualizaciones son su orientación de sistemas, el uso de equipos interdisciplinarios y la aplicación del método científico a problemas de control.

Por su parte, Levin y Kirk Patrick (1983) señalan que la Investigación de Operaciones proporciona a los gerentes bases cuantitativas para la toma de decisiones. La Investigación de Operaciones eleva la habilidad de un gerente para hacer planes a largo plazo y para resolver los problemas diarios de llevar un negocio, una unidad gubernamental o una institución privada. El Investigador de Operaciones recopila e interpreta datos, construye y experimenta con modelos matemáticos, predice operaciones futuras y luego hace recomendaciones, haciendo aplicación del método científico.

Posteriormente, Daellenbach, George y McNickle (1987), en su libro Introducción a las Técnicas de Investigación de consideran que la Investigación de Operaciones trata de un enfoque científico al análisis de muchos tipos de problemas complejos de toma de decisión (económicos, de ingeniería o de medio ambiente) en la forma en que los enfrentan individuos y organizaciones de todos los tipos, ya sean comerciales o benéficas, privadas o gubernamentales. A menudo el problema estudiado implica el diseño y/o la operación de sistemas o partes de sistemas. Se pretende evaluar las consecuencias

probables de las elecciones de decisión, por lo general, bajo condiciones que requieren de la asignación de escasos recursos: fondos, humanos, tiempo o materias primas. El objetivo es mejorar la efectividad del sistemas como un todo.

En la década de los 90 también se proponen conceptualizaciones interesantes, dentro de las cuales cabe señalar la que Ríos Insua (1996), presenta en su obra, Investigación Operativa: Aunque dar una definición buena es siempre difícil, se reproduce con ligeras modificaciones la que da el profesor Johnson de la Operations Research Office de la John Hopkins University: La Investigación Operativa es la predicción y comparación de valores, efectividad y costes de un conjunto de cursos de acción propuestos, en que intervienen sistemas de hombres y máquinas y está basada sobre un modelo descrito mediante una metodología lógica o matemática que permita determinar los valores de los parámetros de los cursos de acción, mediante análisis de observaciones anteriores o de operaciones experimentales convenientemente diseñadas.

Mathur y Solow (1996) consideran que los administradores utilizan las matemáticas y las computadoras para tomar decisiones racionales en la resolución de modelos, para llegar a la solución óptima de problemas que involucran un gran número de alternativas. El estudio de estos diversos métodos y la forma en que los administradores los usan en el proceso de decisión es la esencia de la Investigación de Operaciones. Las técnicas de la Investigación de Operaciones se aplican a las siguientes dos categorías básicas de problemas: determinísticos y estocásticos.

Posteriormente, en la edición de uno de los libros clásicos de Investigación de Operaciones, Taha (2004) considera a la Investigación de Operaciones como una herramienta dominante e indispensable para tomar decisiones. Un elemento principal de la Investigación de Operaciones es el modelado matemático. Aunque la solución del modelo matemático establece una base para tomar una decisión, se deben tener en cuenta factores intangibles o no cuantificables, por ejemplo, el comportamiento humano, para poder llegar a una decisión final.

En otra de las obras clásicas que trata sobre Investigación de Operaciones, Hillier y Lieberman (2006), exponen que el proceso comienza por la observación cuidadosa y la formulación del problema, incluyendo la recolección de los datos pertinentes. El siguiente paso es la construcción de un modelo científico generalmente matemático con el cual se intenta abstraer la esencia del problema real. En esta etapa se propone la hipótesis de que el modelo será una representación tan precisa de las características esenciales de la situación, que permitirá que las conclusiones soluciones, obtenidas sean válidas también para el problema real. Después se llevan a cabo los experimentos adecuados para probar esta hipótesis, para modificarla si es necesario y para verificarla en determinado momento

este paso se conoce como validación del modelo. En cierto sentido, la Investigación de Operaciones involucra la investigación científica creativa de las propiedades fundamentales de las operaciones. Sin embargo, es más que esto. La Investigación de Operaciones se ocupa también de la administración práctica de la organización. Por lo tanto, para tener éxito, también debe proporcionar conclusiones claras que el tomador de decisiones pueda usar cuando sea necesario.

En esta memoria, la Investigación de Operaciones se entenderá como la disciplina que mediante la implementación de modelos matemáticos y algoritmos computacionales, por grupos interdisciplinarios, permite realizar un proceso de toma de decisiones, el cual conduce a la optimización del objetivo previamente definido en organizaciones sistémicas de naturaleza heterogénea en los campos de conocimiento de la Economía, de la Ingeniería, de la Industria, de las Finanzas, etc.

2.5 Naturaleza de la Investigación de Operaciones

La Investigación de Operaciones como herramienta para la toma de decisiones es considerada tanto ciencia como arte. Ciencia por las técnicas y argumentos matemáticos que integra, y arte por que el éxito de las fases que conducen a la solución del modelo matemático depende en gran medida de la creatividad, pericia, juicio técnico, habilidades de comunicación del equipo de investigación.

En este contexto la IO utiliza técnicas de modelamiento matemático, análisis estadístico y optimización matemática, con el objetivo optimizar, es decir de alcanzar soluciones óptimas o cercanas a ellas cuando se enfrentan problemas de decisión complejos. Se espera que las decisiones alcanzadas mediante el uso de un modelo de Investigación Operativa sean significativamente mejores en comparación a aquellas decisiones que se podrían tomar haciendo uso de la simple intuición o experiencia del tomador de decisiones.

El enfoque de la Investigación de Operaciones es la construcción de modelos matemáticos. Un modelo matemático es una herramienta analítica que nos sirve para lograr una visión bien estructurada de la realidad. Así, el propósito del modelo es proporcionar un medio para analizar el comportamiento de las componentes de un sistema con el fin de optimizar su desempeño (identificar el mejor curso de acción posible). Una visión representativa del proceso asociado a la construcción de un modelo de optimización se presenta a continuación:

1. **Definición del problema y recolección de datos**: Se debe definir el problema para el cual se busca proponer un curso de acción. ¿Es un problema relevante? ¿es posible tomar una buena

decisión sin la necesidad de resolver un modelo de optimización? ¿cuáles son sus alcances? ¿cuáles son los factores que influyen en el desempeño del sistema?, etc. La calidad del modelo de optimización dependerá en gran parte de la asertividad en la definición del problema de decisión.

2. **Formulación del modelo matemático**:Un modelo de optimización considera necesariamente una abstracción o simplificación de la realidad. Por un lado se busca que el modelo sea representativo del problema real que se busca representar pero que al mismo tiempo sea simple de modo de favorecer su resolución haciendo uso de un algoritmo ad-hoc. Alcanzar este equilibrio no es trivial. Por ello ante un mismo problema puede existir más de un modelo de optimización que lo represente con distintos niveles de detalle y abstracción.

3. **Solución del modelo**:Una vez construido el modelo de optimización se deben identificar las alternativas de resolución para el mismo. Para ello se puede hacer uso de programas computacionales que utilizan algoritmos de resolución específicos dependiendo de las características del modelo.

4. **Validación**:Se verifica que la solución alcanzada cumpla con las condiciones (restricciones) impuestas al problema.

5. **Implementación y control de la solución**:Una vez verificada la solución se procede a su implementación. Cabe destacar que esto puede dar lugar a actualizaciones del modelo de optimización tanto en términos del modelo como el valor de los parámetros estimados. Por ejemplo, si el modelo de optimización corresponde a un plan maestro para la producción y se genera un cambio en el valor de la hora hombre de los trabajadores será necesario actualizar el valor del parámetro que representa dicho costo para posteriores instancias de resolución.

Como ya se había mencionado anteriormente, la optimización consiste en la selección de una alternativa mejor, en algún sentido, que las demás alternativas posibles. Para esto la Investigación de Operaciones, hace uso de planteamientos de modelos matemáticos que se componen generalmente de los siguientes aspectos:

a. **Función objetivo**: Constituye la medida cuantitativa del funcionamiento del modelo matemático que se pretende optimizar. Como ejemplo de funciones objetivo es posible mencionar: la minimización de los costos de producción, la maximización de los beneficios o utilidad neta de venta de ciertos productos, la minimización del cuadrado de las desviaciones con respecto a unos valores observados, la minimización del material utilizado para la fabricación de un producto, etc.

b. **Variables de decisión**: Manifiestan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Es decir, las variables de decisión son incógnitas que deben ser determinadas a partir de la solución del modelo. Desde un punto de vista funcional se pueden clasificar en variables independientes o principales o de control y variables dependientes o auxiliares o de estado. En el caso de un sistema eléctrico serán los valores de producción de los grupos de generación o los flujos por las líneas. En el caso de la venta, la cantidad de cada producto fabricado y vendido. En el caso de la fabricación de un producto, sus dimensiones físicas.

c. **Las restricciones**: son relaciones entre las variables de decisión y los recursos disponibles. Las restricciones del modelo limitan el valor de las variables de decisión. Se generan cuando los recursos disponibles son limitados. Las restricciones son expresadas mediante ecuaciones e inecuaciones que ciertas variables están obligadas a satisfacer. Por ejemplo, las potencias máximas y mínimas de operación de un grupo de generación, la capacidad de producción de la fábrica para los diferentes productos, las dimensiones del material bruto del producto, etc.

Dicho lo anterior, el proceso de resolver un problema de optimización implica determinar el valor que deben tomar las variables de decisión para hacer óptima la función objetivo, de tal forma que satisfaciendo el conjunto de restricciones.

Al resolver problemas de optimización, en respuesta a la precisión, es posible clasificar los métodos de solución de la siguiente manera:

- Métodos exactos: son algoritmos que siempre devuelven una solución óptima. Los métodos exactos de resolución de problemas se han aplicado con éxito a una cantidad elevada de problemas. Algunos ejemplos de estos métodos son los algoritmos voraces, algoritmos de divide y vencerás, algoritmos de ramificación y poda, backtraking, etc. Todos estos procedimientos resuelven algunos tipos de problemas de forma óptima y en tiempo razonable.

  Sin embargo, existe una clase de problemas, con gran interés práctico para los cuales no se conocen algoritmos que encuentren la solución exacta al problema, o tardaría tanto tiempo en encontrarla que lo hace completamente inaplicable. Además, un algoritmo exacto es completamente dependiente del problema (o familia de problemas) que resuelve, de forma que cuando se cambia el problema se tiene que diseñar un nuevo algoritmo exacto.

- Métodos heurísticos: son algoritmos que producen soluciones sin ninguna garantía de optimalidad y, a su vez, por lo general tienen un tiempo de ejecución mucho menor.

  Para la mayoría de problemas de interés no existe un algoritmo exacto con complejidad polinómica que encuentre la solución óptima a dicho problema. Además, la cardinalidad del

espacio de búsqueda de estos problemas suele ser muy grande, lo cual hace inviable el uso de algoritmos exactos ya que la cantidad de tiempo que necesitaría para encontrar una solución es inaceptable. Debido a estos dos motivos, se necesita utilizar algoritmos aproximados o heurísticos que permitan obtener una solución de calidad en un tiempo razonable.

Un caso particular de los procesos de optimización, es la optimización combinatoria, la cual estudia el modelado y solución algorítmica de problemas donde se busca maximizar (o minimizar) una función de varias variables definidas sobre un conjunto discreto. Algunos problemas clásicos de Optimización Combinatoria son:

- El Problema de la Mochila (Knapsack Problem)
- El Problema del Agente vViajero (Traveling Salesman Problem)
- El Problema de Corte (Cutting Stock Problem)
- El Problema de Cubrimiento de Conjuntos (Set Covering Problem)
- El Problema de Coloreo de Grafos (Graph Coloring Problem)
- El Problema de Transporte (Transportation Problem)
- El Problema de Flujo Máximo (Maximal Flow Problema)

Estos problemas y otros tienen numerosas aplicaciones a problemas que se presentan en la industria, logística, ciencias, ingenierías y en la administración de organizaciones. Como ejemplos podemos mencionar, entre otros, el ruteo y carga de vehículos en redes de distribución, el diseño de redes de telecomunicación, la planificación de la producción, la selección de carteras financieras, la asignación de tareas a procesadores, el análisis de estructuras moleculares, las subastas de frecuencias para radiotransmisión, la asignación de tripulaciones en líneas aéreas, la planificación de la generación de electricidad y la distribución de ambulancias en una región para asegurar un cierto nivel de servicio a su población. La gran variedad de posibles aplicaciones es una fuente constante de problemas nuevos para la investigación en esta área.

2.6 Optimización Combinatoria

La Optimización tiene como objetivo primordial maximizar o minimizar una cierta función objetivo en un cierto dominio x. Las teorías clásicas de Optimización, como por ejemplo el Cálculo Diferencial, Cálculo en Variedades, Teoría de Control Óptimo abordan los problemas cuando el dominio x es

infinito y la función objetivo tiene determinadas características específicas. A su vez, la Optimización Combinatoria es la rama de la Optimización que afronta los problemas con un dominio x finito, conocidos como problemas combinatorios.

Desde la perspectiva de la Matemática clásica, los problemas combinatorios no presentan, en teoría, un alto grado de complejidad puesto que se toma como válida la técnica de la enumeración total, lo cual implica una exploración exhaustiva de todas y cada una de las posibles soluciones. Por otro lado, desde la visión de la Matemática aplicada, cuya preocupación y ocupación reside en dar soluciones a problemas actuales de la Ingeniería, Economía, la Administración, la Industria, el Comercio, etc., los problemas combinatorios tienen poco de triviales, es decir estos suponen enormes retos. De modo que, la enumeración total de 120! posibles permutaciones que pueda seguir un brazo mecánico al tener que perforar 120 puntos distintos de un chasis durante una cadena de montaje, agota sin duda alguna la paciencia de cualquier usuario, aun usando la más potente de las computadoras actuales. La Optimización Combinatoria busca, para cada uno de sus problemas, una mejor alternativa práctica de resolución en contraposición a la normalmente no factible, técnica general de enumeración total, aprovechando para ello la particular estructura combinatoria de cada dominio x.

El génesis histórico de la Optimización Combinatoria se debe a problemas en Economía, relativos a la planificación y administración de operaciones, y el uso eficiente de recursos. Posteriormente se abordaron más aplicaciones técnicas, modelizándose como problemas combinatorios, tales secuenciación de máquinas, planificación de producción, diseño y localización de factorías. En la actualidad los problemas de Optimización Combinatoria están por todas partes: asignación de trabajadores a tareas como pilotos a aviones, embalaje de mercancías, localización de plantas industriales, diseño de campañas de venta e inversión de capital, tamaño de flotas de camiones y planificación del transporte, desarrollo de sistemas de transportes en masas y secuenciación de autobuses y trenes, estudios de códigos genéticos y diseño de nuevas moléculas, asignación controlada de ondas de radio, construcción de compiladores y base de datos, codificación de información, diseño y producción de circuitos VLSI, trazados de redes de comunicación robusta y posicionamiento de satélites. Esta lista podría ser de nunca acabar, incluso en áreas como deportes, arqueología, geografía o psicología se está utilizando la Optimización Combinatoria para responder a importantes cuestiones.

En síntesis, los problemas combinatorios son aquellos en los que se maneja una cantidad finita de posibilidades generadas a partir de un número determinado de elementos. Ahora bien, si lo que deseamos es conocer la mejor manera de dar solución a un problema en vez de hacer frente a todas las posibles soluciones, estaríamos entrando en el mundo de la Optimización Combinatoria.

Todos los problemas de Optimización Combinatoria poseen como común denominador los aspectos siguientes:

- El hecho que tengan un conjunto finito de soluciones implica que las variables sean discretas.
- En determinados casos es difícil encontrar una solución.
- Aunque el conjunto de soluciones candidatas sea finito, este puede llegar a ser tan inmensamente grande, que encontrar una solución se transforma en una ardua tarea.
- Poseen un planteamiento sencillo y mucha dificultad en su resolución.

Como se mencionó anteriormente, el conjunto de posibles soluciones a considerar, aunque sea finito, puede llegar a ser colosal y desde luego el tiempo requerido para encontrar la solución óptima a un problema puede llegar a ser intratable. El tamaño de este conjunto de posibles soluciones estará relacionado al tamaño de los datos de entrada. Así, para el problema de la mochila, se pueden experimentar aumentos en el número de posibilidades a evaluar, por ejemplo, con 4 objetos el número de posibilidades a evaluar serían $2^4 = 16$, mientras que, con 20 objetos, este número de posibilidades aumentaría desorbitadamente hasta $2^{20} = 1048576$.

Complejidad algorítmica

La resolución de los problemas combinatorios cuando el espacio de posibles soluciones es inmenso se realiza mediante la construcción de algoritmos. Desde el punto de vista computacional, es ineludible poseer alguna manera de contrastar una solución algorítmica con otra, para conocer cómo se comporta al momento de ser implementada, especialmente al tratar de resolver problemas grandes.

La complejidad algorítmica es una métrica teórica que se aplica a los algoritmos en este sentido. Es un concepto que fundamental para todos los programadores, pero sin embargo, a menudo se desconoce por completo. En muchos libros se elude el tema porque a menudo se considera difícil de entender.

Saber si un algoritmo es mejor que otro puede estudiarse desde dos puntos de vista: un algoritmo es mejor cuanto menos tarde en resolver un problema, o bien es tanto mejor cuanta menos memoria necesite. A la idea del tiempo que consume un algoritmo para resolver un problema le llamamos complejidad temporal y a la idea de la memoria que necesita el algoritmo le llamamos complejidad espacial. La complejidad espacial, en general, tiene mucho menos interés. El tiempo es un recurso

mucho más valioso que el espacio. (Esto lo podemos ver también en el mundo real: si tienes dinero puedes comprarte una casa más grande, pero no puedes comprarte unos cuantos años más de vida).

A todo esto, la complejidad es una métrica para medir la eficiencia de un algoritmo, valorada generalmente en el peor caso. En lo que respecta a la complejidad en tiempo de un algoritmo, esto es el tiempo que tarda en el peor caso; así mismo la complejidad en espacio de un algoritmo es el espacio que ocupa o usa durante su ejecución en el peor caso. Se pueden realizar comparativas de eficiencia entre algoritmos, teniendo únicamente en cuenta el tamaño de datos de entrada, independientemente de los lenguajes de programación en que están implementados, las máquinas donde son ejecutados, y del valor de sus parámetros de entrada. La eficiencia se expresa mediante una función matemática que dependerá solo del tamaño de entrada. Tenemos los siguientes órdenes de complejidad, según la eficiencia:

- Constante: $O(1)$
- Lineal: $O(n)$
- Logarítmica: $O(\log n)$
- Cuadrada: $O(x^2)$, Cúbica $O(x^3)$, ..., de orden k $O(x^k)$
- Combinación en producto de distintos tipos de complejidades
- Exponencial: $O(x^n)$. por ejemplo: $O(2^n)$, $O(3^n)$
- Factorial: $O(n!)$

Tipos de problemas combinatorios

De acuerdo con la modelización de la solución, es posible clasificar los problemas de la siguiente manera:

- Permutación: Dada una colección de datos, el objetivo es una reordenación de estos. Dentro de este tipo se encuentran problemas como el del agente viajero.
- Binarios: Se busca un subconjunto de elementos dado un conjunto más grande. Las variables de decisión toman valores binarios: "si" o "no", "cierto" o "falso", estos valores representan la pertenencia a la solución por parte de los datos. Dentro de este tipo podemos encontrar problemas como el de la mochila $0-1$.

- Enteros: Se pide ponderar a los elementos del problema, por lo que los valores de las variables representan una cardinalidad en las variables de decisión, es decir, las variables de decisión toman valores enteros, como por ejemplo el problema de la mochila entera.

Métodos de resolución de los problemas combinatorios

A lo largo del tratamiento de los problemas de Optimización Combinatoria, se han venido desarrollando procedimientos para resolver dichos problemas. Sin querer ser exhaustivo, estos se pueden clasificar en algoritmos exactos, heurísticos y metaheurísticos.

Algoritmos exactos

Los algoritmos o métodos exactos por regla general realizan una búsqueda exhaustiva dentro del espacio de todas las soluciones del problema a resolver, lo que conlleva a que su tiempo de ejecución para tamaños de entrada grandes, sea demasiado alto. Pese a esto puede ser una buena idea utilizarlos cuando los tamaños de entrada son reducidos y por supuesto para realizar comparativas con otros métodos aproximados como los heurísticos o metaheurísticos. Como ejemplos de algoritmos exactos es posible mencionar los siguientes: el método de la fuerza bruta o búsqueda exhaustiva, vuelta atrás, Programación Dinámica, Planos de Corte, Branch and Bound.

Algoritmos heurísticos

Las heurísticas son procesos simples que consiguen una buena solución inicial con una complejidad relativamente baja, de manera sencilla y rápida en contrapartida a los métodos exactos, que siempre han de devolver una solución óptima y factible. La mayoría de las heurísticas están basadas en el sentido común, sin seguir un análisis formal de los problemas a resolver. Al ser los algoritmos heurísticos resultado de la inspiración de sus creadores y al intervenir en su diseño diversos elementos, no resulta fácil realizar una clasificación, sin embargo, una clasificación no excluyente de los heurísticos, puede ser esta: heurísticos constructivos, heurísticos de búsqueda (first improvement, best improvement), estrategia voraz, estrategia de descomposición, por ejemplo divide y vencerás, estrategia de reducción, estrategia aleatorizada.

Algoritmos metaheurísticos

Las metaheurísticas, se encuentran en un nivel superior que las heurísticas, guiando su comportamiento, dado que representan una forma más inteligente de resolución de problemas, dando lugar a algoritmos que deben proporcionar soluciones muy cercanas a las óptimas, incluso pudiendo lograrlo. Este término surgió por primera vez en el año 1986 en un artículo de Fred Glover el cual hacía referencia a la búsqueda tabú. Desde ese momento, han surgido multitud de propuestas para diseñar buenos métodos de resolución de problemas.

Las metaheurísticas de manera general se pueden clasificar en trayectorial o poblacional. Ejemplos de trayectoriales son: búsqueda tabú, búsqueda de vecindad variable, GRASP (Greedy Randomized Adaptive Serach Procedures), búsqueda local iterada, recocido simulado. A su vez, ejemplos de metaheurísticas poblacionales son: búsqueda dispersa, reencadenamiento de trayectorias, algoritmos evolutivos (genéticos-GA, meméticos-MA), optimización por colonia de hormigas, inteligencia de enjambres).

2.7 Bibliografía

- Ackoff, R., Sasieni M. (4971). *Fundamentos de Investigación de Operaciones*. México: Limusa - Wiley.
- Bazaraa, S., Javis, J. (1996). *Programación lineal y flujo de redes* . México: Limusa. Quinta edición.
- Daellenbach, J., McNickle, D. (1987). *Introducción a las técnicas de Investigación de Operaciones*. México: Compañía editorial continental.
- Dantzig, G. (1998). *Linear Programming and Extensions*. New York: Princeton University Press.
- Davis M., Aquilano, N., Chase R. *Fundamentos de Dirección de Operaciones*. Madrid: MGraw Hill.
- Faulin, F. (1996). *Notas Sobre Cuestiones Históricas de la Investigación de Operaciones*. España: Editorial Slide Print.
- García, A., Martínez R., Redondo P. (2002). *Métodos de Decisión*. Madrid: Prentice Hall.
- Glover, Fred. Kochenberger G. (2003). *Handbook of Metaheuristics*. E. U. A: Kluwer Academic Publishers.
- Guasch, A., Piera, M., Casanovas, J. (2005). *Modelado y Simulación*. México: Universidad Politécnica de Cataluña. Alfaomega.
- Hillier, F., Lieberman, G. (2006). *Introducción a la Investigación de Operaciones*. México: MGraw-Hill.
- Izar, L. (2008). *Investigación de Operaciones*. México: Editorial Trillas.
- Karmarkar, N. (1984). *A New Polynomial-Time Algorithm for Linear Programming*. Combinatorica, 4: 373 -395.
- Kaufmann A. (1965). *Métodos y Modelos de la Investigación de Operaciones*. México: CECSA. Tomo I.
- Levin, R., Kirkpatrick, C. (1983). *Enfoques cuantitativos a la Administración*. México: Compañía Editorial Continental.
- Martello, S., Toth, P. (1990). *Knapsack Problems*. Great Britain: John Wiley & Sons.

- Mathur, K., Solow D. (1996). *Investigación de Operaciones*. México: Prentice-Hall Hispanoamericana S.A.
- Pérez, I. (1986). *El Algoritmo de Kachiyan*. Revista Lecturas Matemáticas. Sociedad Colombiana de Matemáticas. Volumen VII Números 1 - 2 - 3 . Bogotá.
- Pike, R., Guerra, G. (1989). *Optimización en Ingeniería*. México: Alfaomega.
- Pujolar, M. (2004). *Fundamentos de Programación Lineal y Optimización en Redes*. Barcelona: Universidad Autónoma de Barcelona.
- Ríos, I. (1996). *Investigación Operativa. Programación Lineal y Aplicaciones*. Madrid: Editorial Centro de Estudios Ramón Areces S.A. (CERASA).
- Sala, G. (2001). *Modelización y Optimización*. Valencia: Universidad de Valencia.
- Taha, H. (2004). *Investigación de Operaciones*. México: Pearson Prentice Hall.
- Waddinton, C. (1973). *OR in World War 2 - Operational Research against the U - Boat*. London: Paul Elek Ltd.

CAPÍTULO 3
EL PROBLEMA DE LA MOCHILA

3.1 Introducción

El Problema de la Mochila es un problema simple de entender: hay una persona que tiene una mochila con una cierta capacidad y tiene que elegir que elementos ubicará en ella. Cada uno de los elementos tiene un peso y aporta un beneficio. El objetivo de la persona es elegir los elementos que le permitan maximizar el beneficio sin excederse de la capacidad permitida.

A la vez es un problema complejo, si por complejidad nos referimos a la computacional. Un problema se cataloga como inherentemente difícil si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado. El Problema de la Mochila forma parte de una lista histórica de problemas NP-Completos elaborada por Richard Karp en 1972.

En el caso del Problema de la Mochila, si contáramos con 4 productos, para saber cuál es la mejor solución podríamos probar las $2^4 = 16$ posibilidades. El 2 se desprende del hecho de que cada decisión es incluir o no al producto y el 4 de la cantidad de productos. 16 posibilidades es un número manejable, sin embargo, si la cantidad de elementos por ejemplo ascendiera a 20, tendríamos que analizar nada más y nada menos que $2^{20} = 1048576$ posibilidades.

Como parte de la aplicación del Problema de la Mochila como una forma de emular situaciones reales donde es necesario acomodar artículos de diferentes dimensiones en un espacio reducido. Se puede emplear, como ejemplo, el uso de contenedores en las aduanas, donde se requiere enviar ítems de diferentes pesos, tamaños y valores de beneficio. Por otra parte, en la misma aduana, es necesario almacenar, de manera temporánea, los contenedores mismos, por lo que este problema puede ser resuelto con base en la soluciones propuesta para el problema de la mochila.

En aspectos de criptografía, en el caso de descifrar contraseñas, este problema se puede ver como un número de contenedores que pueden tener n valores cada uno. En otro sentido, cuando es necesario traducir un texto encriptado, en el momento de identificar los espacios, cada palabra puede fungir como un contenedor de n_i ítems (caracteres de la palabra), donde cada caracter i puede tener n posibles artículos.

Como parte de la aplicación del Problema de la Mochila, y haciendo una revisión de la literatura actual se pueden resolver problemas relacionados con:

- La selección de proyectos, donde cada proyecto se puede ver como un contenedor de diferentes tales como: personas, recursos, etc.

- En la solución de problemáticas donde es necesario detectar patrones de corte.
- En situaciones donde se evidencia problemas de distribución de carga (física, eléctrica, etc.).
- Cuando se requiere abastecer vehículos de transporte y entrega de productos de diferentes tamaños que deben ser colocados en múltiples compartimentos de igual o diferente tamaño.
- Asignación de procesadores y datos en sistemas distribuidos.

El Problema de la Mochila se puede resolver por medio de diferentes técnicas, métodos exactos, heurísticos y metaheurísticos. Por ejemplo, mediante el uso de la Programación Dinámica, en donde generalmente se emplean cuatro tipos de visualización: árbol de recursión, grafo de dependencia, tabla de valores y tabla de decisiones. También se pueden usar los denominados algoritmos voraces, con los cuales no siempre es posible dar una solución óptima al problema, sin embargo es posible obtener aproximaciones muy buenas en tiempo razonable. Así mismo se ha implementado una gran variedad de metaheurísticas, entre ellas, metaheurísticas parametrizadas, mediante la combinación de parámetros.

Es importante señalar que existen variantes para el Problema de la Mochila, entre los cuales podemos encontrar:

- Problema de la Mochila 0-1: cuando existe 1 solo objeto por cada tipo.
- Problema de la Mochila no Acotado: Cuando algún tipo contiene un número infinito de objetos.
- Problema de la Mochila de Múltiple Elección: Cuando estamos ante la presencia de un Problema de Mochila 0-1, donde los objetos están subdivididos en clases, y solamente podemos seleccionar un objeto de cada clase.
- Problema de la Mochila Múltiple: cuando en un Problema de Mochila 0-1 se tiene más de una mochila, cada una con su capacidad.

En este apartado se trabajará sobre un tipo de Problema de Mochila 0-1, específicamente el perteneciente a la familia de los problemas de Programación Entera, donde los objetos del conjunto no se repiten y no se puede partir.

3.2 Formulación Matemática del Problema de la Mochila

Supongamos que estamos planeando un viaje de senderismo; y estamos, por lo tanto, interesados en llenar una mochila con los elementos que se consideran necesarios para el viaje. Hay n diferentes tipos de elementos que se consideran deseables; estos podrían incluir una botella de agua, manzana, naranja, sándwich, etc. adelante. Cada tipo de elemento tiene un conjunto dado de dos atributos, es decir, un peso (o volumen) y un valor que cuantifica el nivel de importancia asociado con cada unidad de ese tipo de elemento.

Figura 3.1: Problema de la Mochila

Dado que la mochila tiene un peso (o volumen) limitado por su capacidad, el problema de interés consiste en averiguar cómo cargar la mochila con una combinación de unidades de los tipos especificados de elementos que se obtiene el mayor valor total. Lo que acabamos de describir se conoce como el problema de la mochila.

Una gran variedad de problemas de asignación de recursos se puede convertir en el marco deL Problema de la Mochila. La idea general es pensar en la capacidad de la mochila como la cantidad disponible de un recurso y los tipos de elementos como las actividades a las que este recurso puede ser asignado. Dos ejemplos rápidos son la asignación de un presupuesto de publicidad para las promociones del individuo productos y la asignación de su esfuerzo a la preparación de los exámenes finales en diferentes asignaturas.

Los datos del problema se pueden expresar en términos matemáticos de la siguiente manera:

- Los objetos están numerados por el índice i variando de 1 a n .
- Los números w_i y p_i representan el peso y el valor del número i .
- La capacidad de la mochila se denomina en esta fórmula W .

Existen muchas manera de llenar la mochila, para decidir a cada uno de ellos debemos de decir para cada objeto si lo metemos a la mochila o no, pudiendo utilizar el código binario que cuando $x_i = 1$, metemos el objeto a la mochila, o $x_i = 0$, se pone afuera, y para ir llenando esta mochila podemos

utilizar un vector de contenido, que comprende: $X = (x_1, x_2, ..., x_n)$, entonces podemos expresar una función del contenido del vector.

Para un contenido dado X , el valor total de la bolsa es:

$$Z(x) = \sum_{i=1}^{n} x_i p_i$$

De la misma manera, la suma de los pesos de los objetos es:

$$W(x) = \sum_{i=1}^{n} x_i w_i$$

El problema entonces lo podemos enunciar como un contenido de vectores $X = (x_1, x_2, ..., x_n)$, que tienen componentes de ceros y unos, teniendo como máximo la restricción de la función $Z(x)$.

$$W(x) = \sum_{i=1}^{n} x_i w_i \leq W$$

Esto quiere decir que la suma de los pesos (o sea, la función $W(x)$), de los objetos que pusimos en la mochila no deben de superar la capacidad de esta, (o sea, la W). Podemos decir que:

- $Z(x)$ es una función objetivo (como su nombre lo dice, representa el objetivo del problema, esta expresión se maximiza o se minimiza.
- Un vector X que cumple con la restricción W en la tercera fórmula se le nombra factible (o sea, que se pude hacer).
- Si tenemos un resultado máximo en $Z(x)$, entonces X es óptimo.
- Se le pueden agregar otras restricciones según tengamos un caso, en esta liga, encontramos diferentes casos singulares.

Con esto podemos argumentar que tenemos un problema de decisión cuando para decidir a cada uno de ellos debemos de decir para cada objeto si lo metemos a la mochila o no, que cuando $x_i = 1$, metemos el objeto a la mochila, o $x_i = 0$, se pone afuera y podemos decir que es un problema de optimización porque podemos encontrar funciones objetivo, valores óptimos utilizando las declaraciones matemáticas.

3.3 Tratamiento metodológico

El tratamiento metodológico que se realizará con el Problema de la Mochila es el siguiente: primero se resolverán ciertos problemas de baja dimensión, mediantes diferentes métodos; para iniciar se utilizará un algoritmo exacto (búsqueda exhaustiva), después un algoritmo de Programación Dinámica, seguidamente se propone encontrar la solución con un algoritmo heurístico y luego se confirmarán los resultados obtenidos con la aplicación de un herramienta computacional como lo es el WinQsb.

Ejemplo 3.1 . *Un turista nacional planea salir el fin de semana a la Isla de Ometepe. Hay cuatro artículos que desea llevar consigo, pero entre todos sobrepasan las 5 kilogramos que considera puede cargar.*

El peso y valor de cada artículo se muestran en la siguiente tabla

artículo	**1**	**2**	**3**	**4**
Peso	2	3	4	5
Valor	3	4	5	6

Tabla 3.1: *Datos ejemplo 3.1*

¿Qué artículos tendría que llevar para que el valor de la mochila sea máximo?

Solución 1: (Implementando el algoritmo exacto de búsqueda exhaustiva)

La primera forma para resolver este problema será mediante la implementación del algoritmo de búsqueda exhaustiva. La búsqueda búsqueda exhaustiva, también es conocida búsqueda combinatoria, búsqueda exhaustiva o sencillamente fuerza bruta, es una técnica trivial, que consiste en enumerar sistemáticamente todos los posibles candidatos para la solución de un problema, con el objetivo de determinar si dicho candidato satisface la solución al mismo.

La búsqueda exhaustiva es sencilla de implementar y, siempre que exista, encuentra una solución. Sin embargo, su costo de ejecución es proporcional al número de soluciones candidatas, el cual es exponencialmente proporcional al tamaño del problema. Por el contrario, la búsqueda por fuerza

bruta se usa habitualmente cuando el número de soluciones candidatas no es elevado.

Para resolver el problema primeramente se planteará el modelo matemático asociado a dicho problema:

$$\begin{aligned} &\text{Maximizar} && 3x_1 + 4x_2 + 5x_3 + 6x_4 \\ &\text{Sujeto a} && 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 5 \end{aligned}$$

Seguidamente, para lograr visualizar de una mejor manera la totalidad de posibles opciones de solución, se elabora un diagrama de árbol, el cual facilita la enumeración total de las soluciones posibles:

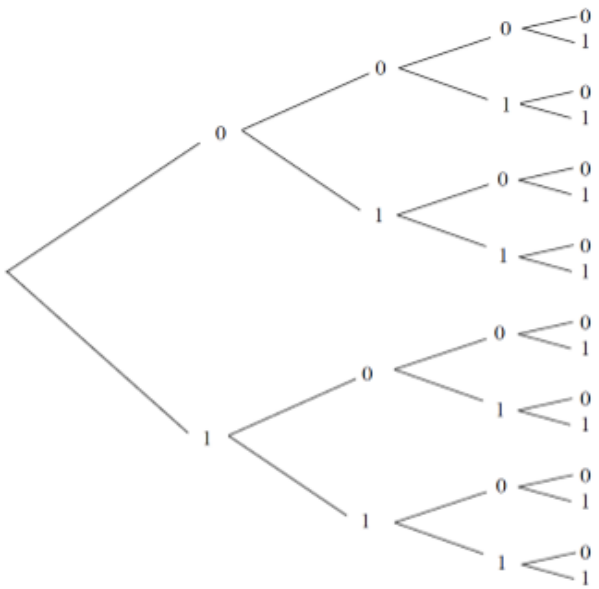


Figura 3.2: Diagrama de árbol

Al recorrer las ramas del árbol se va obteniendo el total de posibles soluciones al problema, las cuales las expresaremos en forma de un vector como el siguiente $[0\ 0\ 0\ 0]$. El valor 0 en una posición significa que ese objeto no se ingresará a la mochila, en cambio el valor 1 indica que el dicho objetos si será parte de la mochila. Por tanto, el vector antes expuesto indica que de los cuatro objetos posibles, a la mochila únicamente se meterá el artículo número 4.

Una vez encontradas y enumeradas todas las posibles soluciones, se procede a determinar con cuál de ellas se obtiene el beneficio máximo, respetando la restricción de volumen. A continuación se muestra una tabla donde se exponen las soluciones, el peso y beneficio asociada a cada una de ellas.

N°	Posible solución	Peso	Valor
1	[0 0 0 0]	0	0
2	[0 0 0 1]	5	6
3	[0 0 1 0]	4	5
4	[0 0 1 1]	9	11
5	[0 1 0 0]	3	4
6	[0 1 0 1]	8	10
7	[0 1 1 0]	7	9
8	[0 1 1 1]	12	15
9	[1 0 0 0]	2	3
10	[1 0 0 1]	7	9
11	[1 0 1 0]	6	8
12	[1 0 1 1]	11	14
13	[1 1 0 0]	5	7
14	[1 1 0 1]	10	13
15	[1 1 1 0]	9	12
16	[1 1 1 1]	4	18

Tabla 3.2: *Datos solución 1*

De todas las posibles opciones de respuestas, sólo algunas de ellas satisfacen (1, 2, 3, 5, 9 y 13) la restricción referida al peso de la mochila y de estas, la opción en la que se obtiene el valor máximo es la número trece , en la cual se incluyen en la mochila los artículos 1 y 2, obteniendo el valor óptimo de 7.

Solución 2: (Implementando un algoritmo exacto de Programación Dinámica)

El objetivo básico en la Programación Dinámica consiste en "descomponer" un problema de optimización en k variables a una serie de problemas con menor número de variables más fáciles de resolver. Precisamente esto es lo que se hace en el Problema de la Mochila; reducir un problema inicial de n variables a n problemas de 1 variable. En este sentido, se podría decir que la Programación Dinámica se basa en un método de descomposición.

La técnica de Programación Dinámica evita explorar todas las secuencias posibles por medio de la resolución de subproblemas de tamaño creciente y almacenamiento en una tabla de las soluciones óptimas de esos subproblemas para facilitar la solución de los problemas más grandes. A contin-

uación se expone el algoritmo para resolver el problema propuesto.

Algoritmo Bottom up Approach
Para encontrar el valor máximo de la mochila:

```
para i=1 hasta n
     dp [i][0] =0

para j=1 hasta w
     dp [0] [j] =0
para i=1 hasta n
para j=1 hasta w
si peso [i] <= W
   dp [i] [j] = max (dp [i-1] [j], dp [i-1] [j-peso[i]] + valor [i])
de lo contrario
   dp [i] [j] = dp [i-1] [j]
Retornar dp [n] [w]
```

Para determinar los objetos que serán parte de la mochila:

```
Mientras i, j >0
si dp [i] [j]  es distinto de dp [i-1] [j] ( el i elemento es parte de
la mochila)
y hacer i=i-1, j=j- peso[i]
de lo contrario
i=i-1 (asume que el elemento i-ésimo no está en la mochila)
```

Mediante la implementación del algoritmo lo que se hace es rellenar la siguiente matriz

N°	**Peso (Volumen)**	**Valor (Beneficio)**
0	Posición	0
1	2	3
2	3	4
3	4	5
4	5	6

Tabla 3.3: *Datos solución 2*

El algoritmo, primeramente lo que hace es rellenar de cero la primera fila (fila 0) y primera columna (columna 0), lo cual significa iniciar con una mochila nula, que tienen un valor asociado de cero. La primera fila y columna se rellena de la siguiente manera:

```
para j=1 hasta 5
     dp [0] [j]=0
para i=1 hasta 4
     dp [i] [0]=0
```

Posteriormente se rellena cada una de las filas, comparando el peso (volumen) de los objetos disponibles con el peso o capacidad de la mochila, si el peso de un objeto disponible es menor o igual que el peso que soporta la mochila, entonces este objeto se introduce en la mochila, y se registra en la matriz el valor asociado a dicho objeto. Por ejemplo, para la fila 1, el peso del primer objeto (2) es mayor que el peso de la mochila (1), esto implica que no es posible introducir este objeto a esa mochila y se procede a registrar cero en la correspondiente posición de la matriz. Luego se compara el peso del mismo objeto con el peso de la siguiente capacidad de la mochila (2), como son iguales entonces se introduce a la mochila el objeto 1 cuyo valor es 3. Mediante la implementación del algoritmo se obtiene lo siguiente:

Para la fila 1:

```
i=1, j=1, entonces  peso [1] >1
          dp[1][1] = dp [0][1]=0

i=1, j=2, entonces peso [1] <= 2
          dp[1][2] = max (dp [0][2], dp [0][2-2] + valor[1])
          = max (0, 0 + 3) =3

i=1, j=3,entonces peso[1] <= 3
         dp[1][3] = max (dp [0][3], dp [0][3-2] + valor[1])
         = max (0, 0 + 3) =3

i=1, j=4, entonces peso[1] <=4
          dp[1][4] = max (dp [0][4], dp [0][4-2] + valor[1])
          = max (0, 0 + 3) =3

i=1, j=5, entonces peso[1] <=5
```

```
          dp[1][5] = max (dp [0][5], dp [0][50-2] + valor[1])
          = max (0, 0 + 3) =3
```

De forma análoga, para la fila 2 tenemos:

```
i=2, j=1, entonces peso [2]>1
          dp[2][1] = dp [1][1]=0

i=2, j=2, entonces peso [2]>2
          dp[2][2] = dp [1][2]=3

i=2, j=3, entonces peso [2]<=3
          dp[2][3] = max (dp [1][3], dp [1][3-3] + valor[2])
          = max (3, 0 + 4) =4

i=2, j=4, peso [2]<=4
          dp[2][4] = max (dp [1][4], dp [1][4-3] + valor[2])
          = max (3, 0 + 4) =4

i=2, j=5, entonces peso [2]<=5
          dp[2][5] = max (dp [1][5], dp [1][5-3] + valor[2])
          = max (3, 3 + 4) =7
```

La siguiente fila se obtiene de la siguiente forma:

```
i=3, j=1, entonces peso [3]>1
          dp[3][1] = dp [2][1]=0

i=3, j=2, entonces peso [3]>2
          dp[3][2] = dp [2][2]=3

i=3, j=3, entonces peso [3]>3
          dp[3][3] = dp [2][3]=4

i=3, j=4, entonces peso [3]  <=4
          dp[3][4] = max (dp [2][4], dp [2][4-4] + valor[3])
          = max (4, 0 + 5) =5
```

```
i=3, j=5, entonces peso [3]<=5
          dp[3][5] = max (dp [2][5], dp [2][5-4] + valor[3])
          = max (7, 0 + 5) =7
```

Para calcular los valores de la última fila se procede de una manera similar:

```
i=4, j=1, entonces peso[4]>1
          dp[4][1] = dp [3][1]=0

i=4, j=2, entonces peso [4]>2
          dp[4][2] = dp [3][2]=3

i=4, j=3 , entonces peso [4]>3
           dp[4][3] = dp [3][3]=4

i=4, j=4 , entonces peso [4]>4
           dp[4][4] = dp [3][4]=5

i=4, j=5 , entonces peso [4]<=5
           dp[4][5] = max (dp [3][5], dp [2][5-5] + valor[4])
           = max (7, 0 + 5) =7
```

Como consecuencia de lo antes expuesto se obtiene la matriz que permite determinar el beneficio máximo que se alcanza con esta mochila, que en este caso corresponde al valor 7, y que es justamente el mismo valor generado mediante el algoritmo de fuerza bruta.

N°	**Peso (Volumen)**	**Valor (Beneficio)**	**0**	**1**	**2**	**3**	**4**	**5**
0	Posición	0	0	0	0	0	0	0
1	2	3	0	0	3	3	3	3
2	3	4	0	0	3	4	4	7
3	4	5	0	0	3	4	5	7
4	5	6	0	0	3	4	5	7

Tabla 3.4: *Datos solución 2*

Es importante señalar que la tabla por sí sola, únicamente nos permite determinar la solución óptima al Problema de la Mochila, pero hasta este momento no se conocen los objetos que se introducirán en

ella. Es justamente, la segunda parte del algoritmo quien nos permitirá solventar esta situación.

Esta segunda parte del algoritmo consiste en comparar el último valor de la matriz con el de la fila anterior ($dp[i][j] \neq dp[i-1][j]$), si son distintos, el objeto i será parte de la mochila y se calcula un nuevo $i(i = i - 1)$, y un nuevo $j(j = j - peso[i])$, si ocurre que son iguales, entonces se descarta el objeto i y se continua comparando, mientras $i, j > 0$.

Este procedimiento se muestra a continuación:

Como $dp[4][5] = dp[3][5]$, entonces el objeto $i = 4$ se descarta, y lo mismo ocurre con los objetos $i = 3$, puesto que ocurre $dp[3][5] = dp[2][5]$, ahora bien, como $dp[2][5] \neq dp[1][5]$, el objeto $i = 2$ será parte de la mochila y se procede a calcular un nuevo j, que en este caso sería $j = 5 - 3 = 2$, luego se continua con la comparación: $dp[1][2] \neq dp[0][2]$, esto hace indicar que el objeto $i = 1$ se deberá incluir en la mochila y hacemos $j = 2 - 2 = 0$. El algoritmo se detiene aquí puesto que debe ser $j > 0$, ante esta situación es posible afirmar que se está ante la presencia del valor óptimo $[1\ 1\ 0\ 0]$, es decir que los objetos que se deben meter a la mochila son el 1 y 2, obteniendo de esta manera un valor máximo de 7.

Solución 3: (Implementando el algoritmo heurístico del coeficiente de rendimiento)

Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. Los algoritmos heurísticos son procedimientos simples basados en el sentido común que se supone ofrecerán una buena solución, aunque no necesariamente la óptima, a problemas difíciles, de un modo fácil y rápido.

Una importante ventaja que presentan las heurísticas respecto a las técnicas que buscan soluciones exactas es que, por lo general, permiten una mayor flexibilidad para el manejo de las características del problema. No suele resultar complejo diseñar algoritmos heurísticos. Además, generalmente ofrecen más de una solución, lo cual permite ampliar las posibilidades de elección del que decide, sobre todo cuando existen factores no cuantificables que no han podido ser añadidos en el modelo, pero que también deben ser considerados. Por otra parte, suele ser más fácil de entender la fundamentación de las heurísticas que los complejos métodos matemáticos que utilizan la mayoría de técnicas exactas.

Una solución intuitiva pero que puede no ser óptima la podemos encontrar con un algoritmo que se implementa de la siguiente forma:

- Ordenar la lista de objetos de forma descendente de acuerdo a la siguiente proporción

$$r_i = \frac{b_i}{v_i}$$

- Seleccionar los objetos hasta llenar la mochila.
- Ahora tenemos dos opciones: si el siguiente objeto ya no cabe completo en la mochila podemos quedarnos con una fracción de él, obteniendo un beneficio igual a

$$\frac{pesototal - pesoactual}{pesoobjeto}$$

  o no meter ningún objeto más en la mochila. La primera opción se elige cuando el problema es de mochila fraccional y la segunda, cuando el problema es de mochila $0/1$.

Al ejecutar el primer paso del algoritmo se obtiene la siguiente tabla:

Objeto	**b**	**v**	**r**
1	3	2	1.5
2	4	3	1.3
3	5	4	1.2
4	6	5	1.2

Tabla 3.5: *Datos solución 3*

Mediante el segundo paso de la heurística mencionada, elegiríamos los productos 1 y 2, ya que al tratar de ingresar el objeto 3 se estaría excediendo la capacidad permitida. Entonces la solución sería $[1\ 1\ 0\ 0]$, generando un beneficio de 7. Se puede notar que, en este caso la solución encontrada por el heurístico coincide con la solución óptima, encontrada primeramente cuando enumeramos todas las posibles opciones y luego al aplicar Programación Dinámica.

Solución 4: (Implementando la herramienta computacional WinQsb)

El WinQsb o simplemente QSB (Quantitative System Business), es un paquete de herramientas muy versátil que contiene 19 módulos, que permiten el análisis y resolución de modelos matemáticos, problemas administrativos, de producción, proyectos, inventarios, transporte, entre muchos otros. Ofrece una interfaz básica pero amigable, es un software de ayuda a la toma de decisiones que contiene herramientas muy útiles para resolver distintos tipos de problemas en el campo de la Investigación de Operaciones.

Uno de estos 19 módulos está referido a la implementación de Programación Dinámica, y dentro de esta se abordan los tres diferentes modelos:

- Problema de la Diligencia (Stagecoach Problem)
- Problema de la Mochila (Snapsack Problem)
- Programación de Producción e Inventarios (Production and Inventory Scheduling)

En el caso del problema que nos ocupa, el Problema de la Mochila, se desarrolla bajo dos consideraciones, primero teniendo en cuenta el peso y luego el volumen. Este es un problema que también podría resolverse por Programación Lineal entera teniendo en cuenta la función objetivo y restricciones siguientes:

Siendo x_j el elemento a transportar. Para el caso del volumen se reformaría la primera restricción cambiando los coeficientes por los volúmenes de los ítems.

Sea j: la variable que representa el artículo:

- $x_{(j)}$:el número de unidades cargadas del artículo j.
- $w_{(j)}$:el espacio o el peso que demanda cada unidad del artículo j.
- $R_{(jX(j)}$:la función del retorno del artículo j si se llevan $x_{(j)}$ unidades en la mochila, del artículo j.
- $g_{(j,w)}$:retorno de total acumulativo dado el espacio w disponible para el artículo j
- $g_{(w)} = máximo\{R_{(jX(j))} + g[i-1, w - W_{(j)X(j)}]\}$

Veamos entonces el procedimiento a seguir para resolver el problema con esta herramienta computacional:

1. Elegir módulo de Programación Dinámica.

Figura 3.3: Módulo del WinQsb

2. Seleccionar nuevo problema y Knapsack Problem

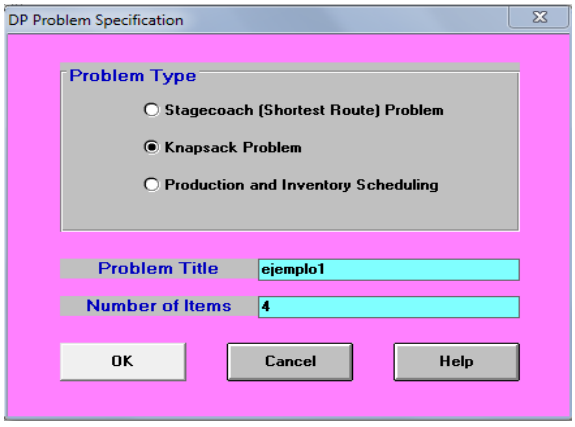


Figura 3.4: Selección del problema

3. Ingresar los datos del problema

Item (Stage)	Item Identification	Units Available	Unit Capacity Required	Return Function (X: Item ID) (e.g., 50X, 3X+100, 2.15X^2+5)
1	X1	M	2	3X1
2	X2	M	3	4X2
3	X3	M	4	5X3
4	X4	M	5	6X4
Knapsack	Capacity =	5		

Figura 3.5: Datos del problema

4. Mostrar la solución del problema

07-20-2017 Stage	Item Name	Decision Quantity (X)	Return Function	Total Item Return Value	Capacity Left
1	X1	1	3X1	3	3
2	X2	1	4X2	4	0
3	X3	0	5X3	0	0
4	X4	0	6X4	0	0
	Total	Return	Value =	7	CPU = 0

Figura 3.6: Solución del problema

La solución nos indica que se deben incluir a la mochila los objetos 1 y 2 los cuales generan un retorno total de 7.

Ejemplo 3.2 *Un naviero tiene un buque carguero con capacidad de hasta 500 toneladas. El carguero transporta contenedores de diferentes pesos para una determinada ruta. En la ruta actual el carguero puede transportar algunos de los siguientes contenedores:*

Contenedor	**1**	**2**	**3**	**4**	**5**
Peso en (cientos de toneladas)	1	2	1	3	4
Valor (miles de dólares)	3	5	4	6	7

Tabla 3.6: *Datos ejemplo* 3.2

El analista de la empresa del armador desea determinar el envío (conjunto de contenedores) que maximiza el valor de la carga transportada.

Solución 1: (Implementando el algoritmo exacto de búsqueda exhaustiva)

El modelo matemático del problema es el siguiente:

$$\begin{aligned} &\text{Maximizar} \quad 3x_1 + 5x_2 + 4x_3 + 6x_4 + 7x_5 \\ &\text{Sujeto a} \quad\quad x_1 + 2x_2 + x_3 + 3x_4 + 4x_5 \leq 5 \end{aligned}$$

N°	Posible solución	Peso	Valor
1	[0 0 0 0 0]	0	0
2	[0 0 0 0 1]	4	7
3	[0 0 0 1 0]	3	6
4	[0 0 0 1 1]	7	13
5	[0 0 1 0 0]	1	4
6	[0 0 1 0 1]	5	11
7	[0 0 1 1 0]	4	10
8	[0 0 1 1 1]	8	17
9	[0 1 0 0 0]	2	5
10	[0 1 0 0 1]	6	12

Tabla 3.7: *Datos solución 1*

N°	Posible solución	Peso	Valor
11	[0 1 0 1 0]	5	11
12	[0 1 0 1 1]	9	18
13	[0 1 1 0 0]	3	9
14	[0 1 1 0 1]	7	16
15	[0 1 1 1 0]	6	15
16	[0 1 1 1 1]	10	22
17	[1 0 0 0 0]	1	3
18	[1 0 0 0 1]	5	10
19	[1 0 0 1 0]	4	9
20	[1 0 0 1 1]	8	16
21	[1 0 1 0 0]	2	7
22	[1 0 1 0 1]	6	14
23	[1 0 1 1 0]	5	13
24	[1 0 1 1 1]	9	20
25	[1 1 0 0 0]	3	8
26	[1 1 0 0 1]	7	15
27	[1 1 0 1 0]	6	14
28	[1 1 0 1 1]	10	21
29	[1 1 1 0 0]	4	12
30	[1 1 1 0 1]	8	19
31	[1 1 1 1 0]	7	18
32	[1 1 1 1 1]	11	25

Tabla 3.8: *Datos solución 1*

De la matriz anterior es posible apreciar que de todas las posibles opciones, la opción número 23, [1 0 1 1 0] es la que satisface la restricción referida al peso y que maximiza el valor de la carga transportada. En este caso los contenedores que deben enviar son el 1, 3 y 4 con lo cual se obtendrá una ganancia de \$ 13000.

Solución 2: (Implementando un algoritmo exacto de Programación Dinámica).

Es importante recordar que con la ejecución de la primer parte del algoritmo el objetivo principal es la construcción de la siguiente matriz

N°	Peso (Volumen)	Valor (Beneficio)	0	1	2	3	4	5
0	Posición	0						
1	1	3						
2	2	5						
3	1	4						
4	3	6						
5	4	7						

Tabla 3.9: *Datos solución 2*

La fila 0 y la columna 0 se rellena de ceros de la siguiente manera:

```
para j=1 hasta 5
     dp[0][j]=0

para i=1 hasta 5
     dp[i][0]=0
```

Posteriormente procedemos a rellenar la fila 1:

```
i=1, j=1, entonces peso[1]<=1
          dp[1][1] = max (dp [0][1], dp [0][1-1] + valor[1])
          = max (0, 0 + 3)=3

i=1, j=2, entonces peso [1]<=2
          dp[1][2] = max (dp [0][2], dp [0][2-1] + valor[1])
          = max (0, 0 + 3)=3

i=1, j=3, entonces peso[1]<=3
          dp[1][3] = max (dp [0][3], dp [0][3-1] + valor[1])
          = max (0, 0 + 3)=3

i=1, j=4, entonces peso[1]<=4
          dp[1][4] = max (dp [0][4], dp [0][4-1] + valor[1])
          = max (0, 0 + 3)=3
```

```
i=1, j=5, entonces peso[1]<=5
          dp[1][5] = max (dp [0][5], dp [0][5-1] + valor[1])
          = max (0, 0 + 3)=3
```

Para la fila 2 tenemos:

```
i=2, j=1, entonces peso [2]>1
          dp[2][1] = dp [1][1]=3

i=2, j=2, entonces peso [2]<=2
          dp[2][2] = max (dp [1][2], dp [1][2-2] + valor[2])
          = max (3, 0 + 5) =5

i=2, j=3, entonces peso [2]<=3
          dp[2][3] = max (dp [1][3], dp [1][3-2] + valor[2])
          = max (3, 3 + 5)=8

i=2, j=4, entonces peso [2]<=4
           dp[2][4] = max (dp [1][4], dp [1][4-2] + valor[2])
           = max (3, 3+5)=8

i=2, j=5, entonces peso [2]<=5
          dp[2][5] = max (dp [1][5], dp [1][5-2] + valor[2])
          = max (3, 3 + 5)=8
```

Para la fila 3 se tiene:

```
i=3, j=1, entonces peso[3]<=1
          dp[3][1] = max (dp [2][1], dp [2][1-1] + valor[3])
          = max (3, 0 + 4)=4

i=3, j=2, entonces peso [3]<=2
          dp[3][2] = max (dp [2][2], dp [2][2-1] + valor[3])
          = max (5, 3 + 4)=7

i=3, j=3, entonces peso [3]<=3
          dp[3][3] = max (dp [2][3], dp [2][3-1] + valor[3])
          = max (8, 5 + 4)=9
```

```
i=3, j=4, entonces peso [3]<=4
          dp[3][4] = max (dp [2][4], dp [2][4-1] + valor[3])
          = max (8, 8 + 4)=12

i=3, j=5, entonces peso [3]<=5
          dp[3][5] = max (dp [2][5], dp [2][5-1] + valor[3])
          = max (8, 8 + 4)=12
```

Para la fila 4:

```
i=4, j=1, entonces peso[4]>1
          dp[4][1] = dp [3][1]=4

i=4, j=2, entonces peso [4]>2
          dp[4][2] = dp [3][2]=7

i=4, j=3, entonces peso [4]<=3
          dp[4][3] = max (dp [3][3], dp [3][3-3] + valor[4])
          = max (9, 0 + 6)=9

i=4, j=4, entonces peso [4]<=4
          dp[4][4] = max (dp [3][4], dp [3][4-3] + valor[4])
          = max (12, 4 + 6)=12

i=4, j=5, entonces peso [4]<=5
          dp[4][5] = max (dp [3][5], dp [2][5-3] + valor[4])
          = max (12, 7 + 6)=13
```

Para la fila 5:

```
i=5, j=1, entonces peso[5]>1
          dp[5][1] = dp[4][1]=4

i=5, j=2, entonces peso [5]>2
          dp[5][2] = dp[5][2]=7

i=5, j=3, entonces peso [5]>3
```

```
          dp[5][3] = dp[4][3]=9

i=5, j=4, entonces peso [5]<=4
          dp[5][4] = max (dp [4][4], dp [4][4-4] + valor[5])
          = max (12, 0 + 7)=12

i=5 j=5, entonces peso [5]<=5
         dp[5][5] = max (dp [4][5], dp [4][5-4] + valor[5])
         = max (13, 4 + 7)=13
```

Finalmente, obtenemos la siguiente matriz:

N°	**Peso (Volumen)**	**Valor (Beneficio)**	**0**	**1**	**2**	**3**	**4**	**5**
0	Posición	0	0	0	0	0	0	0
1	1	3	0	3	3	3	3	3
2	2	5	0	3	5	8	8	8
3	1	4	0	4	7	9	12	12
4	3	6	0	4	7	9	12	13
5	4	7	0	4	7	9	12	13

Tabla 3.10: *Datos solución 2*

Luego aplicamos la parte del algoritmo que nos permite determinar los objetos que serán parte de la mochila:

Como $dp[5][5] = dp[4][5]$, entonces el objeto $i = 5$ se descarta, luego como $dp[4][5] \neq dp[3][5]$, el objeto $i = 4$ será parte de la mochila y se procede a calcular un nuevo j, que en este caso sería $j = 5 - 3 = 2$, luego se continua con la comparación: $dp[3][2] \neq dp[2][2]$, esto hace indicar que el objeto $i = 3$ se deberá incluir en la mochila y hacemos $j = 2 - 1 = 1$, como $dp[2][1] = dp[1][1]$, entonces el objeto $i = 2$ no se incluye, después se observa que $dp[1][1] \neq dp[0][1]$, entonces el objeto $i = 1$ también se mete a la mochila. Al llegar a $j = 0$ el algoritmo se detiene aquí puesto que debe ser $j > 0$, ante esta situación es posible afirmar que se está ante la presencia del valor óptimo $[1\ 0\ 1\ 1\ 0]$, es decir que los objetos que se deben meter a la mochila son el 1, 3 y 4 obteniendo de esta manera un valor máximo de 1300.

Solución 3: (Implementando el algoritmo heurístico coeficiente de rendimiento)

Recordemos que el algoritmo heurístico propuesto funciona de la siguiente manera:

1. Ordenar los objetos de forma descendente de acuerdo al cociente b_i/v_i.
2. Seleccionar los objetos hasta llenar la mochila.
3. Ahora, si el siguiente objeto ya no cabe completo en la mochila, entonces no meter ningún objeto más.

Procediendo a implementar los pasos del algoritmo obtenemos:

Objeto	**b**	**v**	**r**
1	4	1	4
2	3	1	3
3	5	2	2.5
4	6	3	2
5	7	4	1.7

Tabla 3.11: *Datos solución 3*

El segundo paso del algoritmo nos permite tomar la decisión de meter a la mochila los primeros tres productos, ya que el hecho de ingresar el objeto 4 excedería la capacidad permitida. Por tanto la solución sería $[1\ 1\ 1\ 0\ 0]$, generando un beneficio de 13000 dólares. Se puede notar que, en este caso la solución encontrada por el heurístico coincide con la solución óptima, encontrada primeramente cuando enumeramos todas las posibles opciones y luego al aplicar Programación Dinámica.

Solución 4: (Implementando la herramienta computacional WinQsb)

1. Seleccionar un nuevo problema Knapsack Problem.

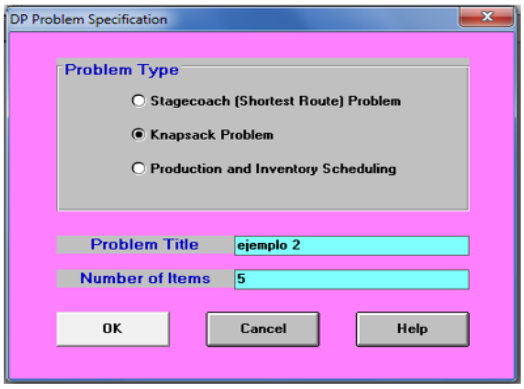


Figura 3.7: Selección del problema

2. Ingresar los datos del problema

Item (Stage)	Item Identification	Units Available	Unit Capacity Required	Return Function (X: Item ID) (e.g., 50X, 3X+100,
1	X1	1	1	3X1
2	X2	1	2	5X2
3	X3	1	1	4X3
4	X4	1	3	6X4
5	X5	1	4	7X5
Knapsack	Capacity =	5		

Figura 3.8: Datos del problema

3. Mostrar la solución del problema

07-20-2017 Stage	Item Name	Decision Quantity (X)	Return Function	Total Item Return Value	Capacity Left
1	X1	1	3X1	3	4
2	X2	0	5X2	0	4
3	X3	1	4X3	4	3
4	X4	1	6X4	6	0
5	X5	0	7X5	0	0
	Total	Return	Value =	13	CPU = 0.02

Figura 3.9: Solución del problema

La solución nos indica que se deben incluir a la mochila los objetos 1, 3 y 4 los cuales generan un retorno total de 13.

Ejemplo 3.3 *Se debe realizar un envío de 10 objetos distintos. El beneficio y volumen aparecen en la siguiente tabla:*

N°	**Beneficio (Dólares)**	**Volumen ($\mathbf{cm^3}$)**
1	70	10
2	90	20
3	90	30
4	80	40
5	100	50
6	80	60
7	90	70
8	150	80
9	130	90
10	140	100

Tabla 3.12: *Datos del ejemplo* 3.3

Determinar el envío máximo que no exceda un volumen de $100\ cm^3$.

Solución 1: (Implementando un algoritmo exacto de Programación Dinámica)

En este caso, se tiene un total de 10 objetos $(n = 10)$ y una restricción de volumen de $80m^3(W = 80)$, para ello modelo matemático asociado al problema es el que sigue:

$$\text{Maximizar} \quad 70x_1 + 90x_2 + 30x_3 + 80x_4 + 100x_5 + 80x_6 + 90x_7 + 150x_8 + 130x_9 + 140x_{10}$$

$$\text{Sujeto a} \quad 10x_1 + 20x_2 + 30x_3 + 40x_4 + 50x_5 + 60x_6 + 70x_7 + 80x_8 + 90x_9 + 100x_{10} \leq 100$$

Resolver este problema mediante el algoritmo de la búsqueda exhaustiva implicaría un espacio de posibles soluciones de $2^{10} = 1024$, es por ello que resulta poco razonable aplicarlo.

Entonces se hace necesario la implementación de otra forma de resolver el problema, a continuación se muestra una:

Con el algoritmo se llena la siguiente matriz

Volumen	**Beneficio**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Posición	0									
10	70									
20	90									
30	90									
40	80									
50	100									
60	80									
70	90									
80	150									
90	130									
100	140									

Tabla 3.13: *Datos solución 1*

Para rellenar la primera columna y primera fila:

```
para j=1 hasta 8
     dp [0] [j] =0

para i=1 hasta 10
```

```
dp [i] [0] =0
```

Para rellenar la primer columna:

```
i=1, j=1, entonces  peso[1]<=10
          dp[1][1] = max (dp [0][1], dp [0][10-10] + valor[1])
          = max (0, 0 + 70)

i=1, j=2, entonces peso [1]<=20
          dp[1][2] = max (dp [0][2], dp [0][20-10] + valor[1])
          = max (0, 0 + 70) =70

i=1, j=3, entonces peso[1]<=30
          dp[1][3] = max (dp [0][3], dp [0][30-10] + valor[1])
          = max (0, 0 + 70) =70

i=1, j=4, entonces peso[1]<=40
          dp[1][4] = max (dp [0][4], dp [0][40-10] + valor[1])
          = max (0, 0 + 70) =70
i=1, j=5, entonces peso[1] <=50
          dp[1][5] = max (dp [0][5], dp [0][50-10] + valor[1])
          = max (0, 0 + 70) =70

i=1, j=6, entonces peso[1]<=60
          dp[1][6] = max (dp [0][6], dp [0][60-10] + valor[1])
          = max (0, 0 + 70) =70

i=1, j=7, entonces peso[1]<=70
          dp[1][7] = max (dp [0][7], dp [0][70-10] + valor[1])
          = max (0, 0 + 70) =70

i=1, j=8, entonces peso[1]<=80
          dp[1][8] = max (dp [0][8], dp [0][80-10] + valor[1])
          = max (0, 0 + 70) =70
```

Para la fila 2 tenemos:

```
i=2, j=1, entonces  peso[2]>10
          dp[2][1] = dp [1][1]=70

i=2, j=2, entonces peso [2] <= 20
          dp[2][2] = max (dp [1][2], dp [1][20-20] + valor[2])
          = max (70, 0 + 90)=90

i=2, j=3, entonces peso [2] <= 30
          dp[2][3] = max (dp [1][3], dp [1][30-20] + valor[2])
          = max (70, 70 + 90)=160

i=2, j=4, entonces peso [2]<= 40
          dp[2][4] = max (dp [1][4], dp [1][40-20] + valor[2])
          = max (70, 70 + 90)=160

i=2, j=5, entonces peso [2]<= 50
          dp[2][5] = max (dp [1][5], dp [1][50-20] + valor[2])
          = max (70, 70 + 90)=160

i=2, j=6, entonces peso [2]<= 60
          dp[2][6] = max (dp [1][6], dp [1][60-20] + valor[2])
          = max (70, 70 + 90)=160

i=2, j=7 , entonces peso [2]<= 70
           dp[2][7] = max (dp [1][7], dp [1][70-20] + valor[2])
           = max (70, 70 + 90)=160

i=2, j=8, entonces peso [2] <= 80
          dp[2][8] = max (dp [1][7], dp [1][80-20] + valor[2])
          = max (70, 70 + 90)=160
```

Las filas que siguen de obtienen de manera similar y a partir de esto obtenemos la siguiente matriz:

Volumen	**Beneficio**	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Posición	0	0	0	0	0	0	0	0	0	0
10	70	0	70	70	70	70	70	70	70	70
20	90	0	70	90	160	160	160	160	160	160
30	90	0	70	90	160	160	180	250	250	250
40	80	0	70	90	160	160	180	250	250	250
50	100	0	70	90	160	160	180	250	250	260
60	80	0	70	90	160	160	180	250	250	260
70	90	0	70	90	160	160	180	250	250	260
80	150	0	70	90	160	160	180	250	250	260
90	130	0	70	90	160	160	180	250	250	260
100	140	0	70	90	160	160	180	250	250	260

Tabla 3.14: *Datos solución 1*

El valor que maximiza la función objetivo se aprecia en la última fila, última columna de la tabla anterior (260) y para determinar cuáles son los objetos que se ingresarán a la mochila se implementa la segunda parte del algoritmo, el cual consiste en comparar el último valor de la matriz con el de la fila anterior $(dp[i][j] \neq dp[i-1][j])$, si son distintos, el objeto i será parte de la mochila y se calcula un nuevo i $(i = i - 1)$, y un nuevo j $(j = j - peso[i])$, si ocurre que son iguales, entonces se descarta el objeto i y se continua comparando, mientras $i, j > 0$. Este procedimiento se muestra a continuación:

Como $dp[10][8] = dp[9][8]$, entonces el objeto $i = 10$ se descarta, y lo mismo ocurre con los objetos $i = 9$, $i = 8$, $i = 7$, $i = 6$, puesto que para cada uno de ellos ocurre $dp[10][8] = dp[9][8]$, $dp[9][8] = dp[8][8]$, $dp[8][8] = dp[7][8]$, $dp[7][8] = dp[6][8]$, $dp[6][8] = dp[5][8]$, ahora bien, como $dp[5][8] \neq dp[4][8]$, el objeto $j = 5$ será parte de la mochila y se procede a calcular un nuevo j, que en este caso sería $j = 8 - 5 = 3$, luego se continua con la comparación: $dp[4][3] = dp[3][3]$, $dp[3][3] = dp[2][3]$ lo que implica que el objeto 4 y 3 no serán parte de la mochila. Procediendo de forma análoga, se tiene que $dp[2][3] \neq dp[1][3]$, luego el objeto 2 se deberá incluir en la mochila y hacemos $j = 3 - 2 = 1$, después se tiene que $dp[1][1] \neq dp[0][1]$, y por tanto el objeto $j = 1$ también será parte de la mochila, además como $i = 0$ el algoritmo se detiene y se está ante la presencia del valor óptimo $[1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$, es decir que los objetos que se deben meter a la mochila son el 1, 2 y 5, obteniendo de esta manera un valor máximo de 260.

Solución 2: (Implementando el algoritmo heurístico del coeficiente de rendimiento):
Al ejecutar el primer paso del algoritmo se obtiene la siguiente tabla:

Objeto	b	v	r
1	70	10	7
2	90	20	4.5
3	90	30	3
4	80	40	2
5	100	50	2
8	150	80	1.8
9	130	90	1.4
10	140	100	1.4
6	80	60	1.3
7	90	70	1.2

Tabla 3.15: *Datos solución 2*

Mediante el segundo paso de la heurística mencionada, elegiríamos los productos 1, 2 y 3, ya que al tratar de ingresar el objeto 4 se estaría excediendo la capacidad permitida. Entonces la solución sería $[1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$, generando un beneficio de 250. Se puede notar que aunque la solución no es la óptima, si es una muy buena aproximación, la cual se obtiene en menos tiempo que el método expuesto en las solución 1.

Solución 3: (Aplicando la herramienta computacional WinQsb)

1. Seleccionar un nuevo problema Knapsack Problem

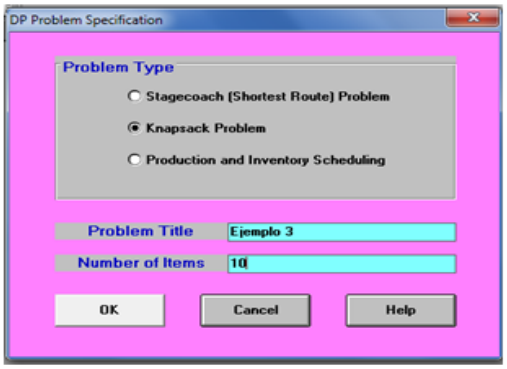


Figura 3.10: Selección del problema

2. Ingresar los datos del problema

Item (Stage)	Item dentification	Units Available	Unit Capacity Required	Return Function (X: Item ID) (e.g., 50X, 3X+100,
1	x1	1	10	70x1
2	x2	1	20	90x2
3	x3	1	30	90x3
4	x4	1	40	80x4
5	x5	1	50	100x5
6	x6	1	60	80x6
7	x7	1	70	90x7
8	x8	1	80	150x8
9	x9	1	90	130x9
10	x10	1	100	140x10
Knapsack	Capacity =	80		

Figura 3.11: Datos del problema

3. Mostrar la solución del problema

07-27-2017 Stage	Item Name	Decision Quantity (X)	Return Function	Total Item Return Value	Capacity Left
1	x1	1	70x1	70	70
2	x2	1	90x2	90	50
3	x3	0	90x3	0	50
4	x4	0	80x4	0	50
5	x5	1	100x5	100	0
6	x6	0	80x6	0	0
7	x7	0	90x7	0	0
8	x8	0	150x8	0	0
9	x9	0	130x9	0	0
10	x10	0	140x10	0	0
	Total	Return	Value =	260	CPU = 0.00

Figura 3.12: Solución del problema

La solución nos indica que se deben incluir a la mochila los objetos 1, 2 y 9 con los cuales se obtiene un valor máximo de 260.

3.4 Problemas propuestos

1. Un barco de 4 toneladas puede cargarse con uno o más de tres artículos. La siguiente tabla da el peso unitario,p_i , en toneladas y el ingreso unitario en miles de dólares, r_i , para el artículo i . El objetivo es determinar la cantidad de unidades de cada artículo que maximizará el rendimiento total.

Artículo i	**Peso (Toneladas)**	**Beneficio ($)**
1	2	31
2	3	47
3	1	14

Tabla 3.16: *Datos del problema 1*

2. Resolver el problema de la mochila para la siguiente instancia:

Objeto	**Peso (grs)**	**Valor ($)**
1	150	20
2	325	40
3	600	50
4	805	36
5	430	25
6	1200	64
7	770	54
8	60	18
9	930	46
10	353	28

Tabla 3.17: *Datos del problema 2*

El peso máximo soportado por la mochila es de $2400grs$.

3. Se debe realizar un envío de 7 objetos distintos. El valor, peso y volumen aparecen en la siguiente tabla:

Objeto	**Valor (Euros)**	**Volumen** $\mathbf{cm^3}$
1	56	21
2	71	16
3	69	17
4	91	28
5	70	12
6	85	31
7	65	9

Tabla 3.18: *Datos del problema 3*

Determinar el envío máximo que no exceda el volumen de $90cm^3$.

4. Una empresa de transporte marítimo de mercancías posee un barco con una bodega cuya capacidad es de $205cm^3$. Se desea transportar cuatro bienes de los que se dispone su volumen y su valor monetario. En la siguiente tabla se muestra dicha información:

Bienes	**Volumen** $(\mathbf{cm^3/Tm})$	**Ingreso ($)**
1	70	1250
2	50	900
3	60	1000
4	75	1200

Tabla 3.19: *Datos del problema 4*

Se trata de determinar la cantidad de $Tm.$ de cada bien que se debe transportar en cada bodega de forma que el ingreso sea máximo.

5. Un camión puede transportar un total de 10 toneladas de productos. Hay cinco clases de productos para transportar, cuyo peso y valor se muestran en la siguiente tabla. Suponiendo que por lo menos se debe transportar un artículo de cada clase, determinar el cargamento que maximiza el valor total.

Clase	**Peso** $(\mathbf{Tm})$	**Valor (miles de dólares)**
A	2	1 000
B	5	2 000
C	6	1800
D	3	900
E	4	2100

Tabla 3.20: *Datos del problema 5*

3.5 Problemas resueltos

Problema 1

Solución 1: (Utilizando el algoritmo heurístico)

Calculamos $r = \frac{beneficio}{peso}$ y ordenamos respecto al cociente:

Artículo i	Peso (Toneladas)	Beneficio ($)	r
2	3	47	15.6
1	2	31	15.5
3	1	14	14

Tabla 3.21: *Datos del problema 1*

Con base a los resultados obtenidos, se toma la decisión de incluir en el barco, solamente el artículo 2, puesto que al tratar de meter el artículo 2 estaríamos violando la restricción referida al peso. Y con esto se obtiene un beneficio de $47 000.

Solución 2: (Utilizando el algoritmo de búsqueda exhaustiva)

Ahora resolveremos el problema mediante la implementación del algoritmo de búsqueda exhaustiva. En la siguiente tabla se muestran el total de posibles soluciones:

Objeto	Posible solución	Peso	Valor
1	[0 0 0]	0	0
2	[0 0 1]	1	4
3	[0 1 0]	3	47
4	[0 1 1]	4	61
5	[1 0 0]	2	31
6	[1 0 1]	3	45
7	[1 1 0]	5	78
8	[1 1 1]	6	92

Tabla 3.22: *Datos del problema 1*

La solución óptima es cargar los artículos 2 y 3, generando con esto un valor máximo óptimo de $61 000. Por tanto podemos apreciar, que con esta instancia, el heurístico genera un valor por debajo del

óptimo, aunque la solución se encontró con un menor esfuerzo en términos de tiempo.

Problema 2

Solución 1: (Utilizando el algoritmo heurístico)

Calculamos $r = \frac{Valor}{peso}$ y ordenamos respecto al cociente:

Objeto	Peso (grs)	Valor ($)	r
8	60	18	0.3
1	150	20	0.1333
2	325	40	0.1230
3	600	50	0.0833
10	353	28	0.0793
7	770	54	0.0701
5	430	25	0.0581
6	1200	64	0.0533
9	930	46	0.0494
4	805	36	0.0447

Tabla 3.23: *Datos del problema 2*

De la tabla anterior podemos deducir que los objetos que serán parte de la mochila son: 8, 1, 2, 3 y 10, los cuales generan un valor de $184.

Problema 3

Solución 1: (Implementando el algoritmo heurístico)

Calculamos $r = \frac{Valor}{volumen}$ y ordenamos respecto al cociente:

Objeto	Valor ($)	Volumen (cm^3)	r
7	65	9	7.6666
5	70	12	5.8333
2	71	16	4.4375
3	69	17	4.0588
4	91	28	3.25
6	85	31	2.7419
1	56	21	2.6666

Tabla 3.24: *Datos del problema 3*

Luego, los objetos que se deben enviar son: 7, 5, 2, 3 y 4 los cuales generan un valor de $366.

Problema 4

Solución 1: (Implementando el algoritmo heurístico)

Calculamos $r = \frac{Ingreso}{volumen}$ y ordenamos respecto al cociente:

Bienes	Volumen (m^3)	Ingreso ($)	r
2	50	900	18
1	70	1 250	17.8571
3	60	1 000	16.6666
4	75	1 200	16

Tabla 3.25: *Datos del problema 4*

Solución 2: (Implementando el algoritmo de búsqueda exhaustiva)

Implementando el algoritmo de búsqueda exhaustiva obtenemos la siguiente tabla en donde se aprecia el total de posibles soluciones:

N^0	**Posible solución**	**Peso**	**Valor**
1	[0 0 0 0]	0	0
2	[0 0 0 1]	75	1 200
3	[0 0 1 0]	60	1 000
4	[0 0 1 1]	135	2 200
5	[0 1 0 0]	50	900
6	[0 1 0 1]	125	2 100
7	[0 1 1 0]	110	1 900
8	[0 1 1 1]	185	3 100
9	[1 0 0 0]	70	1 250
10	[1 0 0 1]	145	2 450
11	[1 0 1 0]	130	2 250
12	[1 0 1 1]	205	3 450
13	[1 1 0 0]	120	2 150
14	[1 1 0 1]	195	3 350
15	[1 1 1 0]	180	3 150
16	[1 1 1 1]	255	4 350

Tabla 3.26: *Datos del problema 4*

La solución óptima es transportar los artículos 1, 3 y 4, generando con esto un valor máximo óptimo de $3 450.

Problema 5
Solución 1: (Utilizando el algoritmo heurístico)

Clase	**Peso (Tn)**	**Valor ($)**	**r**
E	4	2100	525
A	2	1 000	500
B	5	2 000	400
D	3	900	300
C	6	1800	300

Tabla 3.27: *Datos del problema 5*

Con base a los resultados de la tabla anterior, los productos que se deben incluir en el camión son el E, A, y B, con esto el ingreso máximo será de $ 5 100.

Solución 2: (Implementando el algoritmo de búsqueda exhaustiva)

Implementando el algoritmo de búsqueda exhaustiva obtenemos la siguiente tabla en donde se aprecia el total de posibles soluciones:

N^0	**Posible solución**	**Peso**	**Valor**
1	[0 0 0 0 0]	0	0
2	[0 0 0 0 1]	4	2 100
3	[0 0 0 1 0]	3	900
4	[0 0 0 1 1]	7	3 000
5	[0 0 1 0 0]	6	1 800
6	[0 0 1 0 1]	10	3 900
7	[0 0 1 1 0]	9	2 700
8	[0 0 1 1 1]	13	4 800
9	[0 1 0 0 0]	5	200
10	[0 1 0 0 1]	9	4 100
11	[0 1 0 1 0]	8	2 900
12	[0 1 0 1 1]	12	5 000
13	[0 1 1 0 0]	11	3 800
14	[0 1 1 0 1]	15	5 900
15	[0 1 1 1 0]	14	4 700
16	[0 1 1 1 1]	18	6 800
17	[1 0 0 0 0]	2	1 000
18	[1 0 0 0 1]	6	3 100
19	[1 0 0 1 0]	5	1 900
20	[1 0 0 1 1]	9	4 000
21	[1 0 1 0 0]	8	2 800
22	[1 0 1 0 1]	12	4 900
23	[1 0 1 1 0]	11	3 700
24	[1 0 1 1 1]	15	5 800

Tabla 3.28: *Datos del problema 5*

N^0	**Posible solución**	**Peso**	**Valor**
25	[1 1 0 0 0]	7	3 000
26	[1 1 0 0 1]	11	5 100
27	[1 1 0 1 0]	10	3 900
28	[1 1 0 1 1]	14	6 000
29	[1 1 1 0 0]	13	4 800
30	[1 1 1 0 1]	17	6 900
31	[1 1 1 1 0]	16	5 700
32	[1 1 1 1 1]	20	7 800

Tabla 3.29: *Datos del problema 5*

Igual que en la solución 1, los productos que se deben incluir en el camión son: A, B y E, con lo que se obtendrá un ingreso óptimo de $ 5 100.

3.6 Bibliografía

- Babaioff, R. et Al. (2007). *A Knapsack Problem with Applications*. University of Southern California.
- Bazaraa, S., Javis, John J. (1996). *Programación lineal y flujo de redes*. México: Limusa. Quinta edición.
- Bellman R., (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.
- Dorta, I., León, C., Rodríguez, C., Rodríguez, G., y Rojas, A. (2003). *Complejidad Algorítmica: de la Teoría a la Práctica.* III Jornadas de Enseñanza Universitaria de Informática.
- Fayard, D. and Plateau G. (1994). *An exact algorithm for the O-l collapsing knapsack problem*. Discrete Applied Mathematics.
- Fernández, L. Velázquez A. (2009). *Estudio sobre la visualización de las Técnicas de Diseño de Algoritmos*. XII Congreso Internacional de Interacción Persona ? Ordenador (Interacción 2011). Universidad Politécnica de Madrid.
- Gilmore, P.C., & Gomory, R.E., (1966). *The theory and computation of knapsack functions, Operations Research* 14, 1045-1074.
- J. Dréo, A. Pétrowski, y E. Taillard. (2006). *Metaheuristics for Hard Optimization*. Berlin: Springer.
- Karp, R. (1972). *Reducibility Among Combinatorial Problems*. En R.E. Miller y J. W. Thatcher (editories). Complexity of Computer Computations. Nueva York: Plenum.
- Martello, S., Toth, P. (1990). *Knapsack Problems*. Great Britain: John Wiley & Sons.
- Mathur, K., Solow D. (1996). *Investigación de Operaciones*. México: Prentice-Hall Hispanoamericana S.A.
- Pérez, I. (1986). El Algoritmo de Kachiyan. Revista Lecturas Matemáticas. Sociedad Colombiana de Matemáticas. Volumen VII Números 1 - 2 - 3 . Bogotá.
- N. Martí Oliet, Y. Ortega Mallén, A. Verdejo. (2013). *Estructuras de datos y métodos algorítmicos*. España: Garceta.
- Pisinger, D. (2003). Where are the hard knapsack problems?. Technical Report 2003/8, DIKU, University of Copenhagen, Denmark.

- Prawda, J. (1995). *Métodos y modelos de investigación de operaciones, vol. I Modelos Determinísticos*. España.
- Velasco, J. (2010). *NP-Completeness: Complejidad del problema de la Mochila*. Presentación de Tesis. Facultad de Ingeniería Mecánica y Eléctrica. División de Posgrado en Ingeniería en Sistemas. Universidad Autónoma de Nuevo León.

CAPÍTULO 4

EL PROBLEMA DE LA RUTA MÁS CORTA

4.1 Introducción

El Problema de la Ruta más Corta es uno de los problemas más importante y por tanto muy estudiado de optimización combinatoria. Dado los nodos de una red, el problema consiste en encontrar la ruta más corta entre dos nodos de la red, en la cual cada arco tiene un costo (o longitud) no negativo. El objetivo es minimizar el costo (tiempo o longitud) total.

El Problema de la Ruta más Corta tiene muchas aplicaciones, tanto directas como subrutinas en otros algoritmos de optimización. Los algoritmos para este tipo de problemas han sido estudiados desde la década de la década de los 50 y continúan siendo un área activa de investigación. De hecho, ha sido el objetivo de una investigación extensiva durante muchos años y ha dado como resultado la publicación de un gran número de documentos científicos.

El Problema de la Ruta más Corta es fundamental en muchas áreas, como son: investigación de operaciones, ciencia de la computación e ingeniería. Algunas de las razones son:

- La amplia variedad de aplicaciones prácticas como es el envío de algún material entre dos puntos específicos de la forma más eficiente, económica o rápida.
- Existen métodos de solución eficientes, los cuales al ser aplicados a una red con características específicas (acíclica y con costos no negativos), proveen una solución exacta a un tiempo y costo razonables.
- Se puede utilizar como inicio en el estudio de modelos complejos de redes, esto es, cuando no se conoce la estructura de la red se pueden aplicar algoritmos para conocer algunas características de la red (presencia de ciclos negativos).

El Problema de Ruta más Corta tiene muchas aplicaciones prácticas, algunas son: encontrar la ruta más corta o más rápida entre dos puntos en un mapa, redes eléctricas, telecomunicaciones, transporte, planeación de tráfico urbano, trasbordo, diseño de rutas de vehículos, planeación de inventarios, administración de proyectos, planeación de producción, horarios de operadores telefónicos, diseño de movimiento en robótica, redes de colaboración entre científicos, reemplazo de equipo, etc.

Los algoritmos de solución para el Problema de la Ruta más Corta pueden adaptarse en la búsqueda inicial de una solución aproximada de problemas complejos, esto significa que la aplicación consiste precisamente en proveer estructura para varios problemas de optimización combinatoria como: el problema de la mochila, secuencia de alineación en biología molecular (secuenciación del ADN), el problema del agente viajero, etc.

El Problema de la Ruta más Corta se puede resolver mediante la aplicación de algoritmos como:

- Algoritmo de Dijkstra, resuelve el problema de los caminos más cortos desde un único vértice origen hasta todos los otros vértices del grafo.

- Algoritmo de Bellman-Ford, resuelve el problema de los caminos más cortos desde un origen si la ponderación de las aristas es negativa.

- Algoritmo de Floyd-Warshall, resuelve el problema de los caminos más cortos entre todos los vértices.

- Algoritmo de Johnson, resuelve el problema de los caminos más cortos entre todos los vértices y puede ser más rápido que el de Floyd-Warshall en grafos de baja densidad.

- Algoritmo de Viterbi, resuelve el problema del camino estocástico más corto con un peso probabilístico adicional en cada vértice.

- Algoritmo de Búsqueda Anchura, resuelve el problema de los caminos más cortos entre un par de vértices usando la heurística para intentar agilizar la búsqueda.

4.2 Formulación Matemática del Problema de la Ruta más Corta

Dada una red dirigida $G = (V, A, c)$, donde V es el conjunto de nodos, A el conjunto de aristas, c el conjunto de pesos o costos. Se denota por $(i, j) \in A$, el arco que conecta al nodo i con el nodo j, y el costo positivo asociado es c_{ij}. La red tiene dos nodos específicos: el nodo fuente s y el nodo destino t.

El problema consiste en encontrar la ruta (p) más corta (o de costo mínimo) iniciando en el nodo fuente y terminando en el nodo destino, considerando que cada arco (i, j) tiene un costo asociado c_{ij}, es decir, se busca minimizar la función:

$$c(p) = \sum_{(i,j)\in p} c(i, j)$$

Desde la perspectiva de Programación Lineal el Problema de la Ruta más Corta se puede plantear como el envío de una unidad de flujo del nodo origen 1 al nodo destino t, al mínimo costo. Esto es, $b_i = 1$, $b_t = -1$, y $b_i = 0$, para $i \neq 1$ ó t. Entonces, el planteamiento es como sigue:

$$\text{Minimizar} \quad \sum_{j=1}^{t}\sum_{j=1}^{t} c_{ij}x_{ij}$$

$$\text{s.a} \qquad \sum_{j=1}^{t} x_{ij} - \sum_{j=1}^{t} x_{ki} = y_j = \begin{cases} 1 & \text{si } i = 1 \\ 0 & \text{si } i \neq 1 \text{ ó } t \\ -1 & \text{si } i = t \end{cases}$$

$$x_{ij} = 0, \quad \text{ó} \quad i, j = 1, 2, ..., t$$

Sin embargo, como las ecuaciones de conservación de flujo son unimodulares, es decir, si existe una solución óptima el método simplex obtendrá valores $1, 0$. Por esta razón la última restricción puede plantearse como: $x_{ij} \geq 0$, $i, j = 1, 2, ..., t$.

Por otro lado, las diferentes formas que puede presentar el Problema de la Ruta más Corta son:

- Del nodo fuente s al nodo destino t. Para que exista solución se debe cumplir:
  1. Existe al menos una trayectoria entre s y t.
  2. No existen circuitos negativos tales que haya una ruta de s a algún nodo del circuito y otra de algún nodo del circuito a t.
- Del nodo fuente s a todo nodo de la red i. Para que exista solución se debe cumplir:
  1. Existen rutas de s a i.
  2. No existen circuitos negativos en la red.
- Entre todo par de nodos. Para que exista solución se debe cumplir:
  1. Existe, al menos, una trayectoria entre todo par de nodos.
  2. No existen circuitos negativos en la red.

En lo que respecta a los tipos de problemas, las rutas a encontrar pueden ser:

- formadas por un simple arco, (s,t), o
- la ruta del nodo s al nodo t que atraviesa por otros nodos.

Además de forma independiente de la variación del problema, se pueden tener los siguientes casos:

- El tipo más sencillo del problema de la ruta más corta es cuando la longitud de cada arco es 1. Esto significa que la longitud de la ruta es exactamente el número de arcos que contiene.

- Cuando todos los arcos tienen distancias (costos) no negativas y no existen circuitos en la red.

- Cuando no existen ciclos dirigidos.

- Cuando no existen ciclos dirigidos con longitudes negativas.

- Ruta más corta en gráficas no dirigidas con longitudes de los arcos no negativas. En este caso, se reemplaza cada arco no dirigido uv por dos arcos dirigidos uv y vu, con la misma longitud que uv.

- Cuando existen arcos con costos negativos. Actualmente no se conoce un algoritmo que resuelva este tipo de problema polinomialmente, y la teoría de la complejidad computacional parece indicar que no existe un algoritmo.

- En gráficas no dirigidas con al menos un arco con longitud negativa. En este caso, al reemplazar el arco uv por y vu, juntos forman un ciclo.

4.3 Tratamiento metodológico

El Problema de la Ruta más Corta lo abordaremos mediante la resolución de problemas que tienen asociado grafos con pocos nodos. El primer problema los resolveremos aplicando el algoritmo de Dijkstra, en cambio el segundo problema utilizaremos el algoritmo de Bellman-Ford y luego para la resolución del tercero implementaremos el algoritmo de Floyd-Warshall. Cabe señalar que con la intención de confirmar los resultados encontrados con cada uno de los algoritmos mencionados, cada problema también lo resolveremos mediante la aplicación del WinQsb.

Ejemplo 4.1 *Una empresa nacional de servicio de televisión por cable necesita determinar la o las trayectorias bajo las cuales se deben extender los cables para conectar los negocios y residencias que demandan el servicio. Lo que el gerente de la empresa quiere es que se inicie el tendido en el lugar 1 y finalice en el sitio 7, de tal manera que monto por el gasto de cables sea mínimo. El grafo que se muestra a continuación representa las localidades y el gasto correspondiente donde se debe realizar la instalación.*

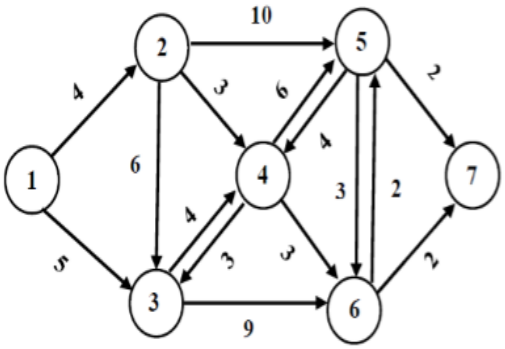


Figura 4.1: Grafo del ejemplo 4.1

Como se puede apreciar la situación antes expuesta representa un Problema de la Ruta más Corta, y para ello, primeramente aplicaremos el algoritmo de Dijkstra.

Solución 1: (Implementación del algoritmo de Dijkstra)

Previamente a la solución del problema presentamos aspectos generales de cómo funciona el algoritmo de Dijkstra:

Teniendo un grafo dirigido ponderando de N nodos no aislados, sea x el nodo inicial, un nodo vector D de tamaño N guardará al final del algoritmo las distancias desde x al resto de los nodos.

1. Inicializar todas las distancias en D con un valor infinito ya que son desconocidas al principio, exceptuando la de x que se debe colocar en 0 debido a que la distancia de x a x sería 0.
2. Sea $a = x$ (tomamos a como nodo actual).
3. Recorremos todos los nodos adyacentes de a, excepto los nodos marcados, llamaremos a estos nodos no marcados v_i.
4. Para el nodo actual, calculamos la distancia tentativa desde dicho nodo a sus vecinos con la siguiente fórmula: $dt(v_i) = D_a + (a, v_i)$. Es decir, la distancia tentativa del nodo v_i es la distancia que actualmente tiene el nodo en el vector D más la distancia desde el dicho el nodo "a" (el actual) al nodo v_i. Si la distancia tentativa es menor que la distancia almacenada en el vector, actualizamos el vector con esta distancia tentativa. Es decir: si $dt(v_i) < D_{vi} \Rightarrow D_{vi} = dt(v_i)$

5. Marcamos como completo el nodo a.

6. Tomamos como próximo nodo actual el de menor valor D (puede hacerse almacenando los valores en una cola de prioridad) y volvemos al paso 3 mientras existan nodos no marcados.

Una vez terminado el algoritmo, D estará completamente lleno. El pseudocódigo del algoritmo es el que se muestra a continuación:

Algoritmo de Dijkstra
Dijkstra (G, s)
Inicializar
for cada v perteneciente a $V[G]$
do $d[v] =$ infinito
$p[v] =$ nulo
$d[s] = 0$
$S =$ vacío
$Q = V[G]$
mientras Q no vacío
do $u =$ nodo v con $min\ d[v]$
$S = S$ unión u se añade al conjunto de nodos finalizados
for cada v perteneciente Adyacente u
Relajación
if $d[v] > d[u] + w(u, v)$ then
$d[v] = d[u] + w(u, v)$
$p(v) = u$

Tabla 4.1: *Pseudocódigo del algoritmo de Dijkstra*

Ahora procederemos a realizar cada una de las iteraciones correspondientes al algoritmo de Dijkstra.

Iteración 0

$[0, -], k = 1, u_1 = 0$

Dado que iniciamos en el nodo 1, en la iteración 1 se marca como permanente dicho nodo, y luego se procede a etiquetar con $[0, -]$ en la cual el 0 indica que hasta ese momento no existe distancia acumulada y símbolo $-$ señala que no existe un nodo inmediato anterior al nodo origen. Gráficamente tendríamos:

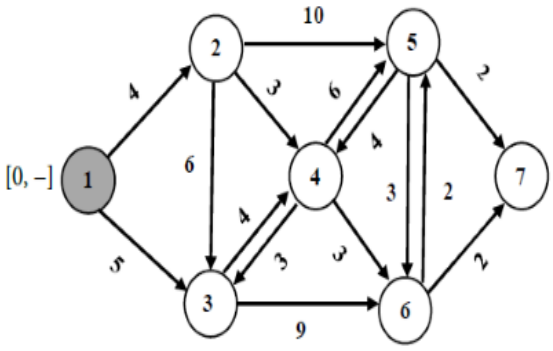


Figura 4.2: Grafo iteración 0

Notamos que los nodos adyacentes al nodo 1 son el nodo 2 y 3, y es por ello que en la siguiente iteración, es decir en la iteración 1 debemos realizar los siguientes cálculos:

Iteración 1

$P = 1, \quad T = \{2, 3, 4, 5, 6, 7\}$
$\Gamma(k) = \Gamma(1) = \{2, 3\}$
$u_2 = u_1 + c_{12} = 0 + 4 = 4 \longrightarrow [u_2, 1] = [4, 1]$
$u_3 = u_1 + c_{13} = 0 + 5 = 5 \longrightarrow [u_3, 1] = [5, 1]$
$min\{u_2, u_3\} = min\{4, 5\} = 4 \longrightarrow permanente[4, 1], k = 2$

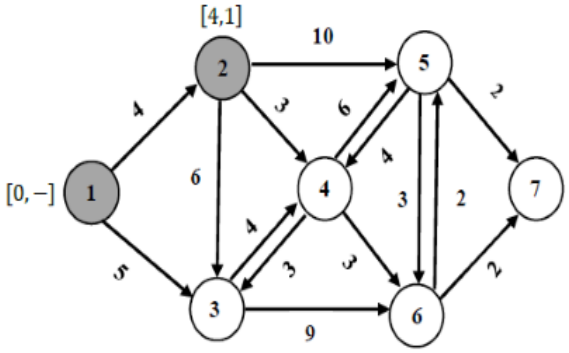


Figura 4.3: Grafo iteración 1

Los resultados obtenidos en esta iteración nos indican que tenemos que etiquetar de forma permanente al nodo 2 con $[4, 1]$ puesto que a este nodo la menor distancia es de 4 procediendo del nodo 1.

Ahora, los nodos adyacentes al nodo 2 son los nodos 3, 4 y 5 y los cálculos correspondientes se muestran en la siguiente iteración:

Iteración 2

$P = 1, 2, \quad T = \{3, 4, 5, 6, 7\}$
$\Gamma(k) = \Gamma(2) = \{3, 4, 5\}$
$u_3 = min\{u_1 + c_{13}, u_2 + c_{23}\} = min\{0 + 5, 4 + 6\} = 5, \longrightarrow [u_3, 1] = [5, 1]$
$u_4 = min\{u_2 + c_{24}, u_3 + c_{34}\} = min\{4 + 3, \infty\} = 4, \longrightarrow [u_4, 2] = [7, 2]$
$u_5 = min\{u_2 + c_{25}, u_4 + c_{45}, u_6 + c_{65}\} = min\{4 + 10, \infty, \infty\} = 14 \longrightarrow [u_5, 2] = [14, 2]$
$min\{u_3, u_4, u_5\} = min\{5, 7, 14\} = 5 \longrightarrow permanente[5, 1], k = 3$

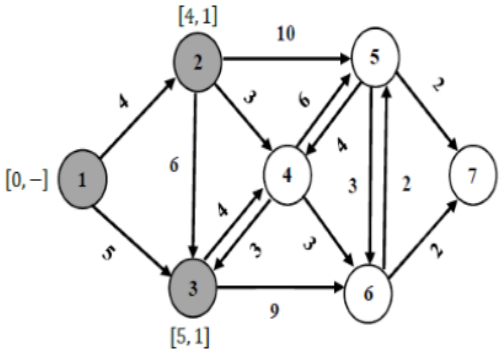


Figura 4.4: Grafo iteración 2

Iteración 3

$P = 1, 2, 3, \quad T = \{4, 5, 6, 7\}$
$\Gamma(k) = \Gamma(3) = \{4, 6\}$
$u_4 = min\{u_2 + c_{24}, u_3 + c_{34}\} = min\{4 + 3, 5 + 4\} = 7, \longrightarrow [u_4, 2] = [7, 2]$
$u_6 = min\{u_3 + c_{36}, u_4 + c_{46}\} = min\{5 + 9, \infty\} = 14 \longrightarrow [u_6, 3] = [14, 3]$
$min\{u_4, u_6\} = min\{7, 14\} = 7 \longrightarrow permanente[7, 2], k = 4$

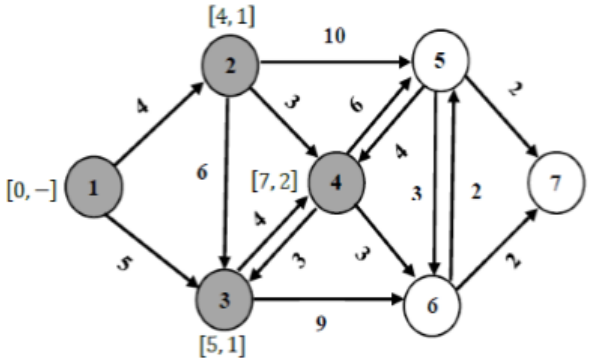


Figura 4.5: Grafo iteración 3

Como resultado de la iteración 3, ahora corresponde etiquetar permanentemente al nodo 4 con $[7, 2]$, es decir que al nodo 4 la menor distancia acumulada es de 7 procediendo del nodo 2.
Seguidamente podemos ver que estando en el nodo 4, los nodos adyacentes a él son los nodos 5 y 6 y al calcular las correspondientes distancias acumuladas tenemos:

Iteración 4

$P = 1, 2, 3, 4, \quad T = \{5, 6, 7\}$
$\Gamma(k) = \Gamma(4) = \{5, 6\}$
$u_5 = min\{u_2 + c_{25}, u_4 + c_{45}, u_6 + c_{65}\} = min\{4 + 10, 7 + 6, \infty\} = 13, \longrightarrow [u_5, 4] = [13, 4]$
$u_6 = min\{u_3 + c_{36}, u_4 + c_{46}, u_5 + c_{56}\} = min\{5 + 9, 7 + 3, \infty\} = 10 \longrightarrow [u_6, 4] = [10, 4]$
$min\{u_5, u_6\} = min\{13, 10\} = 7 \longrightarrow permanente[10, 4], k = 5$

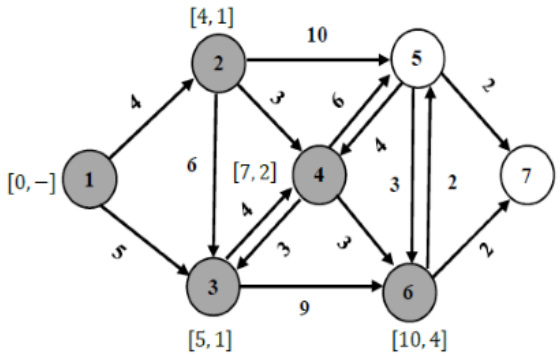


Figura 4.6: Grafo iteración 4

Esta vez, los resultados obtenidos en la iteración 4 nos sugieren que debemos etiquetar con carácter permanente al nodo 6 con $[10, 4]$, es decir que hasta el nodo 6, la menor distancia acumulada es 10,

viniendo del nodo 4. Luego procedemos a realizar la siguiente iteración:

Iteración 5

$P = 1, 2, 3, 4, 6, \quad T = \{5, 7\}$
$\Gamma(k) = \Gamma(6) = \{5, 7\}$
$u_5 = min\{u_2 + c_{25}, u_4 + c_{45}, u_6 + c_{65}\} = min\{4 + 10, 7 + 6, 10 + 2\} = 12, \longrightarrow [u_5, 6] = [12, 6]$
$u_7 = min\{u_5 + c_{57}, u_6 + c_{67}\} = min\{\infty, 10 + 2\} = 12 \longrightarrow [u_7, 6] = [12, 6]$
$min\{u_5, u_7\} = min\{12, 12\} = 12 \longrightarrow permanente[12, 6], k = 5$

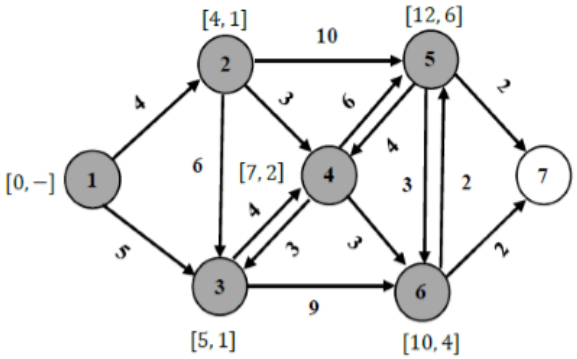


Figura 4.7: Grafo iteración 5

De la misma manera en que veníamos procediendo en las iteraciones anteriores, en la iteración 5, calculamos las distancias acumuladas hasta los nodos adyacentes al 6, es decir, determinamos las distancias al nodo 5 y 7, y con base a los resultados, etiquetamos de manera permanente al nodo 5 con $[12, 6]$, esto nos indica que la distancia acumulada es de 12 viniendo 6. Luego pasamos a implementar la iteración 6:

Iteración 6

$P = 1, 2, 3, 4, 6, 5, \quad T = \{7\}$
$\Gamma(k) = \Gamma(5) = \{7\}$
$u_7 = min\{u_5 + c_{57}, u_6 + c_{67}\} = min\{12 + 2, 10 + 2\} = 12 \longrightarrow [u_7, 6] = [12, 6]$
$min\{u_6\} = min\{12\} = 12 \longrightarrow permanente[12, 6], k = 7$

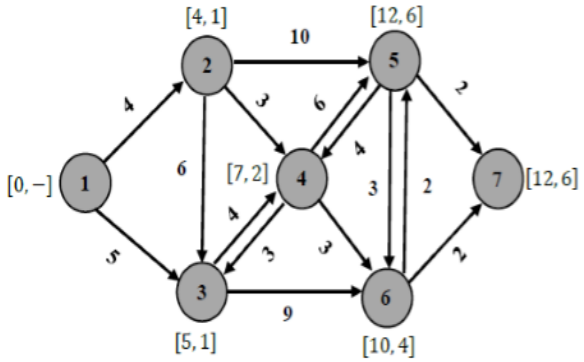


Figura 4.8: Grafo iteración 6

Luego de las seis iteraciones hemos llegado a la conclusión que la ruta más corta y por tanto la que permite utilizar la menor cantidad de cable es la siguiente:

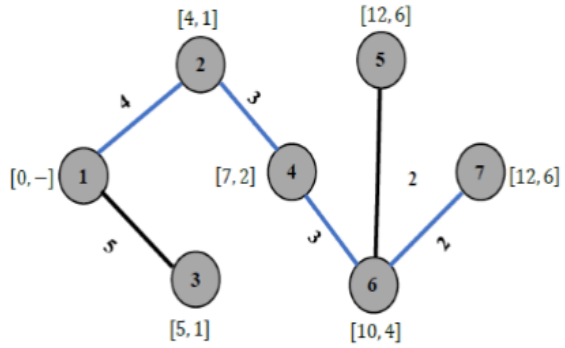


Figura 4.9: Ruta más corta del ejemplo 4.1

Por tanto, la ruta más corta se forma de la siguiene manera:

$$① \longleftarrow [4,1] \longleftarrow ② \longleftarrow [7,2] \longleftarrow ④ \longleftarrow [10,4] \longleftarrow ⑥ \longleftarrow [12,6] \longleftarrow ⑦$$

Solución 2: (Implementando la herramienta computacional WinQsb)

A continuación resolveremos este Problema de la Ruta Más Corta mediante el uso de la herramienta computacional WinQsb. Recordemos que con esto, la idea central es la confirmación de los resultados encontrados por medio del algoritmo de Dijkstra:

a. Elegir módulo de modelado de red.

Figura 4.10: Módulo del Modelo de Redes del WinQsb

b. Seleccionar nuevo problema y Shortest Path Problem

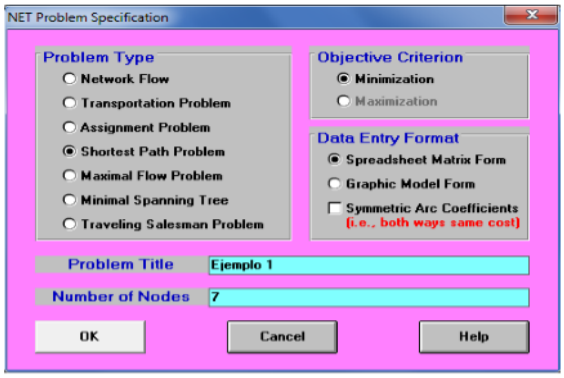


Figura 4.11: Especificaciones del ejemplo 4.1

c. Ingresar los datos del problema

From \ To	Node1	Node2	Node3	Node4	Node5	Node6	Node7
Node1		4	5				
Node2			6	3	10		
Node3				4		9	
Node4					6	3	
Node5						3	2
Node6							2
Node7							

Figura 4.12: Instancia del ejemplo 4.1

c. Mostrar la solución del problema

	From	To	Distance/Cost	Cumulative Distance/Cost
1	Node1	Node2	4	4
2	Node2	Node4	3	7
3	Node4	Node6	3	10
4	Node6	Node7	2	12
	From Node1	To	=	12
	From Node1	To	=	4
	From Node1	To	=	5
	From Node1	To	=	7
	From Node1	To	=	13
	From Node1	To	=	10

Figura 4.13: Solución del ejemplo 4.1

Al finalizar las iteraciones el programa muestra la tabla de resultados en donde se aprecia cada iteración, así como la ruta más corta entre el nodo origen y cada uno de los demás nodos. En el caso particular, la distancia que se deseaba encontrar, es decir la distancia más corta entre el nodo 1 y el nodo 7 es de 12, lo cual viene a confirmar los resultados que habíamos encontrado con el algoritmo de Dikjstra.

Ejemplo 4.2 *Una empresa de comunicaciones desea saber cuál es la distancia más corta entre cada una de las ciudades para formar una red de comunicación. Entonces, dada la siguiente red, lo que se quiere es calcular la distancia más corta entre cada par de ciudades. A continuación se presenta el grafo asociado a la situacion antes expuesta*

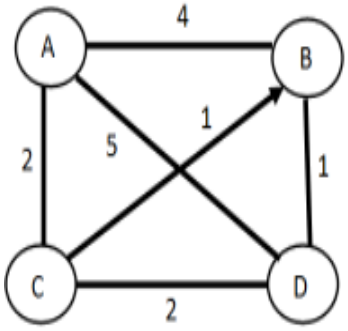


Figura 4.14: Grafo del ejemplo 4.2

Solución 1: (Implementación del algoritmo de Floyd-Warshall)

El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica, teniendo en cuenta que este tipo de programación tiene como fin encontrar una solución óptima a dicho problema recursivamente. A continuación se muestra el pseudocódigo del algoritmo:

Algoritmo Floyd-Warshall
Floyd-Warshall (G)
Inicializar $D = A$ matriz de distancias = matriz de arcos
si $i = j$ o $Dij =$ infinito entonces $Pi, j =$ nulo sino $Pi, j = i$ matriz de caminos
for $k = 1$ to V for $i = 1$ to V for $j = 1$ to V $Di, j = min(Di, j, Di, k + Dk, j)$ si $min = Di$,$k + Dk$,j entonces $Pi, j = Pk, j$
fin

Tabla 4.2: *Pseudocódigo del algoritmo de Dijkstra*

Para iniciar la implementación del algoritmo procedemos a crear dos matrices 4×4, una para las distancias y otra para los recorridos:

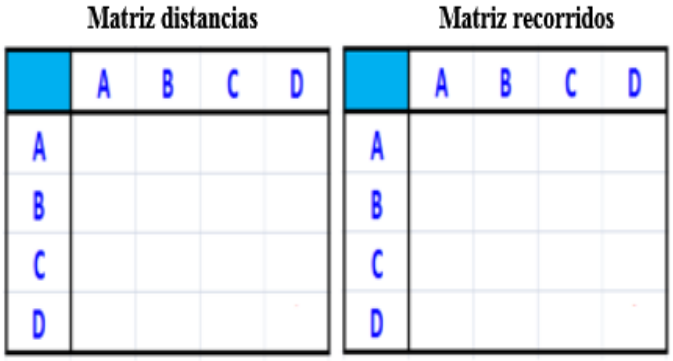
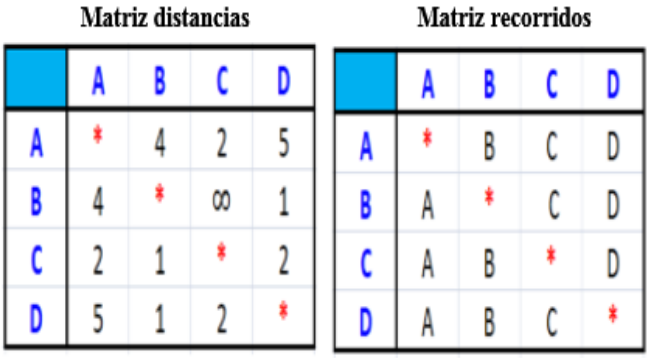


Figura 4.15: Tabla inicial del algoritmo

Despues pasamos a llenar la matriz distancia y matriz recorridos. Para la primera, la diagonal siempre va vacía y el resto de casillas se llenan de acuerdo a las distancias del grafo, el simbolo ∞ indica que el nodo B y C no tienen una conexión directa. En la matriz de recorridos, la diagonal siempre va vacía y las demás casillas se llenan con el mismo número de columna, esto quedaría de la siguiente manera:



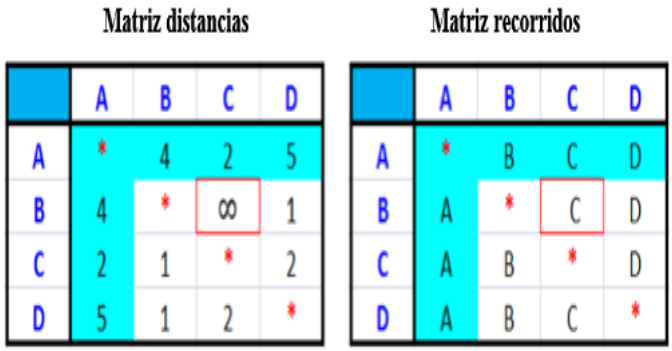
	A	B	C	D
A	*	4	2	5
B	4	*	∞	1
C	2	1	*	2
D	5	1	2	*

	A	B	C	D
A	*	B	C	D
B	A	*	C	D
C	A	B	*	D
D	A	B	C	*

Figura 4.16: Matriz de distancia y matriz de recorrido

Dado que la matrices cuadradas en cuestión son de orden 4, entonces se realizarán cuatro iteraciones.

Iteración 1. Anular la primera fila y la primera columna



	A	B	C	D
A	*	4	2	5
B	4	*	∞	1
C	2	1	*	2
D	5	1	2	*

	A	B	C	D
A	*	B	C	D
B	A	*	C	D
C	A	B	*	D
D	A	B	C	*

Figura 4.17: Iteración 1

Una vez seleccinada la primer fila y la primer columna pasamos a sumar cada elemento de la fila con cada elemento de la columna. Por ejemplo, la posición BD las componentes son 4 y 5. Entonces, $4 + 5 = 9$, ¿9 es menor que 1? No, entonces no se puede mejorar y se pasa a la siguiente posición. Luego, en la posición CD las componentes son 2 y 5, entonces $2 + 5 = 7$, como 7 no es menor que 2, entonces no se puede mejorar y pasamos a la siguiente posición. De la misma manera, en la posición

BC, las componentes son 4 y 2. Luego, $4 + 2 = 6$, como 6 es menor que ∞, entonces se puede mejorar y cambiamos el ∞ por el 6, en la matriz distancias, y en la matriz recorridos, justamente en la misma posición cambiamos la C por la letra de la columna seleccionada en ese momento, es decir por A. Estos cambios se evidencia en la matriz de la siguiente iteración.

Iteración 2. Anular la segunda fila y la segunda columna

Matriz distancias

	A	B	C	D
A	*	4	2	5
B	4	*	6	1
C	2	1	*	2
D	5	1	2	*

Matriz recorridos

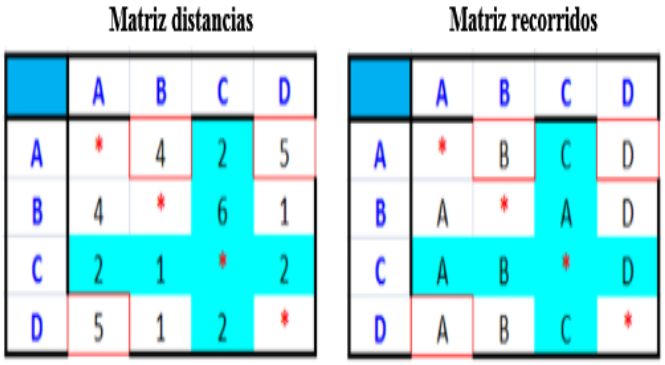
	A	B	C	D
A	*	B	C	D
B	A	*	A	D
C	A	B	*	D
D	A	B	C	*

Figura 4.18: Iteración 2

Por ejemplo, en la posición AC, las componentes son 4 y 6, al realizar la suma correspondiente tenemos que 10 no es menor que 2 y por tanto no se puede mejorar y nos trasladamos a la siguiente posición. Para AD las componentes son 1 y 4, al sumar obtenemos 5 el cual no es menor que 5, y como no podemos mejorar continuamos con la posición CA, para la cual las componentes son 4 y 1. Entonces, como 5 no es menor que 2, no es posible mejorar y continuamos con la posición DA, donde las componentes son 4 y 1, aquí obtenemos 5 como suma y dado que no podemos mejorar pasamos a la posicion DC con componentes son 1 y 6, de esto obtenemos una suma de 7, que no es menor que 2 y desde luego no no se puede mejorar. Después, estando en la posición CD, las componentes son 1 y 1, teniendo a 2 como suma vemos que no es posible mejorar y nos traladamos a la siguiente posición.

Como ya no hay valores libres que se puedan mejorar, entonces no se hace ningún cambio en las matrices durante segunda iteración. Se pasa a la tercera iteración.

Iteración 3. Anular la tercera fila y la tercera columna



	A	B	C	D
A	*	4	2	5
B	4	*	6	1
C	2	1	*	2
D	5	1	2	*

	A	B	C	D
A	*	B	C	D
B	A	*	A	D
C	A	B	*	D
D	A	B	C	*

Figura 4.19: Iteración 3

Procediendo de forma análoga, es decir hay que analizar los valores que quedan en blanco (los que no fueron coloreados) y la idea es mejorar ese valor. Esto se hace cuando la suma de componentes es menor al valor que no está coloreado.

Por ejemplo, en la posición AD, las componentes son 2 y 2. Entonces, como 4 es menor que 5 actualizamos la matriz distancia reemplazando el 5 por el 4, y en la matriz de recorridos cambiamos D por la C. Lo mismo ocurre en la posición AB y DA donde la suma de las componentes es 3 y 4 respectivamente, luego actualizamos, en esas posiciones a la matriz distancias, con dichos valores y también, en la matriz de recorridos cambiamos por C la B y la A. En la otras posiciones de las matrices los valores quedan igual. Cabe señalar que los cambios antes mencionados se visualizan en las matrices que se presentan en la proxima iteración.

Iteración 4. Anular la cuarta fila y la cuarta columna

	A	B	C	D
A	*	3	2	4
B	4	*	6	1
C	2	1	*	2
D	4	1	2	*

	A	B	C	D
A	*	C	C	C
B	A	*	A	D
C	A	B	*	D
D	C	B	C	*

Figura 4.20: Iteración 4

Utilizando el mismo criterio que en las iteraciones anteriores, al única posición que se debe actualizar es la BC, ya que la suma 3 de las componentes es menor que el valor 6. Como no hay más valores que se puedan mejorar, procedemos a exponer la matriz final, tanto de distancias, como recorridos.

Matriz distancias

	A	B	C	D
A	*	3	2	4
B	4	*	3	1
C	2	1	*	2
D	4	1	2	*

Matriz recorridos

	A	B	C	D
A	*	C	C	C
B	A	*	D	D
C	A	B	*	D
D	C	B	C	*

Figura 4.21: Matriz final

Con estas matrices podemos saber las distancias y saber por qué nodo pasa el recorrido en el grafo. Por ejemplo:

- La distancia entre el nodo A y B es 3 y para llegar es preciso recorrer los nodos A-C-B.
- La distancia entre el nodo A y C es 2 y para llegar es necesario recorrer los nodos A-C. Luego,
- La distancia entre el punto A y D es 4, para llegar se requiere recorrer los puntos A-C-D.
- La distancia entre el punto B y A es 4. Para llegar es necesario recorrer los puntos 2-1.
- La distancia entre el punto B y C es 3. Para llegar es necesario recorrer los puntos B-D-C.
- La distancia entre el punto B y D es 1. Para llegar es necesario recorrer los puntos B-D.
- La distancia entre el punto C y A es 2. Para llegar es necesario recorrer los puntos C-A.
- La distancia entre el punto C y B es 1. Para llegar es necesario recorrer los puntos C-B.
- La distancia entre el punto C y D es 2. Para llegar es necesario recorrer los puntos C-D.
- La distancia entre el punto D y A es 4. Para llegar es necesario recorrer los puntos D-C-A.
- La distancia entre el punto D y B es 1. Para llegar es necesario recorrer los puntos D-B.
- La distancia entre el punto D y C es 2. Para llegar es necesario recorrer los puntos D-C.

Solución 2: (Implementando la herramiento computacional WinQsb)

Nuevamente utilizaremos la herramienta computacional WinQsb para resolver el problema del ejemplo 6.2. Cabe señalar que la idea fundamental con esto, es la comprobación de los resultados obtenidos con el algoritmo de Floyd-Warshall:

a. Elegir módulo de modelado de red.

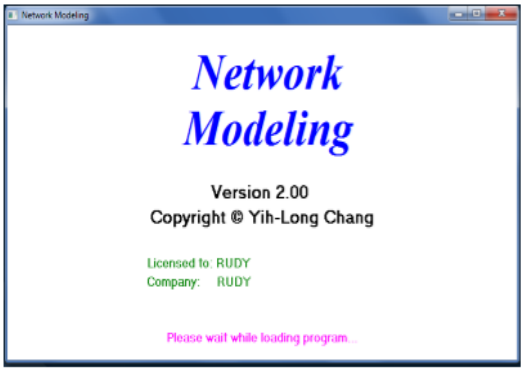


Figura 4.22: Módulo de Modelo de Redes del WinQsb

b. Seleccionar nuevo problema y Shortest Path Problem.

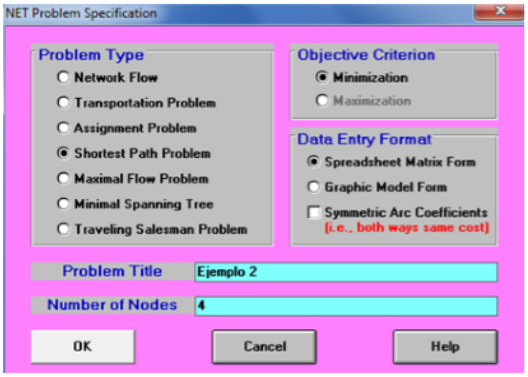


Figura 4.23: Especificaciones del ejemplo 4.2

c. Ingresar los datos del problema

From \ To	Nodo A	Nodo B	Nodo C	Nodo D
Nodo A		4	2	5
Nodo B	4			1
Nodo C	2	1		2
Nodo D	5	1	3	

Figura 4.24: Solución del ejemplo 4.2

d. Mostrar la solución del problema

	From	To	Distance/Cost	Cumulative
1	Nodo A	Nodo C	2	2
2	Nodo C	Nodo D	2	4
	From Nodo A	To Nodo D	=	4
	From Nodo A	To Nodo B	=	3
	From Nodo A	To Nodo C	=	2

Figura 4.25: Solución del ejemplo 4.2

Luego de dos iteraciones el WinQsb muestra en la matriz de resultados la ruta más corta entre el nodo origen y cada uno de los demás nodos. Asi mismo, particularmente, podememos apreciar que la distancia más corta entre el nodo A y el nodo D es de 4 unidades y la ruta es $A - C - D$. Lo antes expuesto nos permite lo cual viene a confirmar los resultados que encontrados con el algoritmo de Floyd-Warshall, el cual fue corrido a mano.

Ejemplo 4.3 *Dado el siguiente grafo, lo que se quiere es calcular la distancia más corta entre cada par de nodos.*

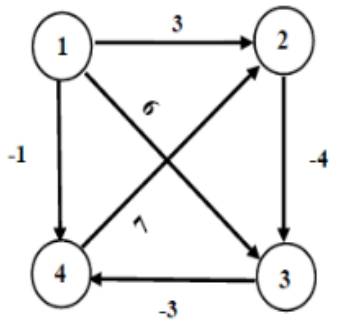


Figura 4.26: Grafo del ejemplo 4.3

Solución 1: (Implementando el algoritmo de Floyd-Warshall)

El tercer algoritmo para resolver el problema de la ruta más corta es el conocido como algoritmo de Bellman-Ford. Este algoritmo viene a solventar la imposibilidad del algoritmo de Dijkstra de lidiar con grafos en los que algunos arcos cuentan con pesos negativos. El de Bellman-Ford es capaz de detectar ciclos negativos y avisar de su existencia evitando entrar en un bucle en el que siempre se puede mejorar el camino más corto. Al igual que el de Dijkstra, se basa en un mecanismo en el que la aproximación a la distancia entre nodos se va mejorando sucesivamente hasta que se alcanza una solución óptima, pero se diferencia en la forma de llevarlo a cabo.

El algoritmo de Bellman-Ford empieza inicializando a 0 la distancia asociada a la fuente y a infinito las distancias acumuladas del resto de nodos. Posteriormente, lleva a cabo el proceso de relajación iterando en todos los nodos distintos de la fuente, comprobando en cada arco que sale de él si la distancia hacia otro nodo es mayor que la suma de la distancia acumulada del primero más el peso del arco que los une, en caso afirmativo establece este resultado como la distancia acumulada por el nodo de llegada y guarda el nodo de partida como su predecesor.

Al final termina comprobando si se han encontrado ciclos negativos, revisando arco por arco si se puede reducir la distancia acumulada del nodo de llegada. Si no se encuentra con uno, el resultado final es un árbol de expansión del camino más corto. El pseudocódigo del algoritmo es el siguiente:

Algoritmo del Vecino más Cercano

Entrada: $G = (V, A); D; s \in V$ $A^T \longleftarrow \emptyset$
$W_{(1:|V|)} \longleftarrow \infty; W_s \longleftarrow 0$
$P \longleftarrow \emptyset$
para cada nodo $i \in V \backslash \{s\}$ hacer
  para cada arco $(i, j) \in A$ hacer
    si $W_j > W_i + d_{ij}$ entonces
    $W_j \longleftarrow W_i + d_{ij}$
    $P_j \longleftarrow i$
    fin si
  fin para
fin para
para cada arco $(i, j) \in A$ hacer
    si $W_j > W_i + d_{ij}$ entonces
    no hay solución
  fin si fin para
para cada nodo $j \in V\{s\}$ hacer
  $i \longleftarrow P_j$
  Seleccionar arco $(i, j) \in A$
  $A^T \longleftarrow A^T \cup \{(i, j)\}$
fin para
$V^T \longleftarrow V$
Salida:$(V^T, A^T); W$

Tabla 4.3: *Pseudocódigo del algoritmo de Bellman-Ford*

Al comenzar a implementar el algoritmo partimos del conjunto de nodos, el conjunto de arcos a formar, el nodo fuente o de partida que es el nodo 1 y la matriz de distancias: Además tenemos que

$$V = \{1, 2, 3, 4\}$$
$$A = \{(1, 2), (1, 3), (1, 4)(2, 3), (2, 4), (3, 4)\}$$

$$s = 1$$
$$D = \begin{bmatrix} * & 5 & 6 & -1 \\ \infty & * & -4 & \infty \\ \infty & \infty & * & -3 \\ \infty & 2 & \infty & * \end{bmatrix}$$

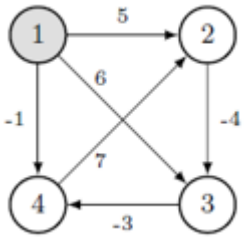


Figura 4.27: Grafo

inicializamos a 0 la distancia asociada a la fuente y a infinito las distancias acumuladas del resto de nodos. Esto es:

$A^T \longleftarrow \phi$
$W_{(1:|V|)} \longleftarrow \infty = \{\infty, \infty, \infty, \infty\}; W_s \longleftarrow 0; W = \{0, \infty, \infty, \infty\}$
$P_{(1:|V|)} \longleftarrow 0$ para cada nodo $i \in V \setminus \{s\}$ hacer

Una vez inicializado el algoritmo, pasamos a realizar cada una de las iteraciones:

Iteración1

Para cada arco $(i, j) \in A$ hacer
Arco$(1, 2)$:
Si $W_j > W_i + d_{ij}$, entonces
$W_2 \longleftarrow W_1 + d_{12} = 0 + 5 = 5$
$P_2 \longleftarrow 1 = \{0, 1, 0, 0\}$
fin si

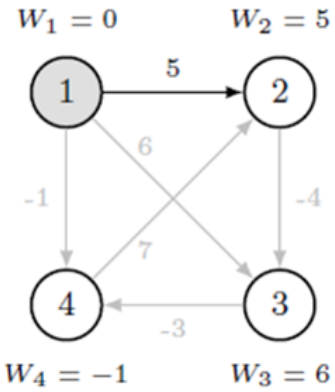


Figura 4.28: Grafo iteración 1

Arco$(1, 3)$:
Si $W_j > W_i + d_{ij}$, entonces
$W_3 \longleftarrow W_1 + d_{13} = 0 + 6 = 6$
$P_3 \longleftarrow 1 = \{0, 1, 1, 0\}$
fin si

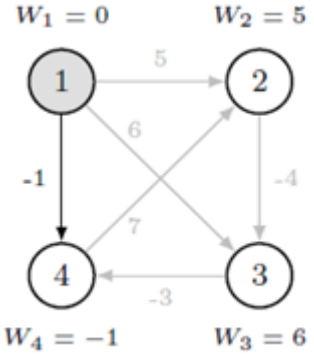


Figura 4.29: Grafo iteración 1

Arco$(1,4)$:
Si $W_j > W_i + d_{ij}$, entonces
$W_4 \longleftarrow W_1 + d_{14} = 0 + (-1) = -1$
$P_4 \longleftarrow 1 = \{0, 1, 1, 1\}$
fin si
fin para

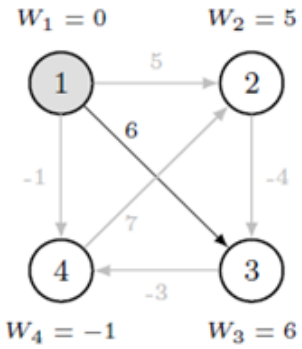


Figura 4.30: Grafo iteración 1

Iteración2

Para cada arco $(i, j) \in A$ hacer
Arco$(2,3)$:
Si $W_j > W_i + d_{ij}$, entonces
$W_3 \longleftarrow W_2 + d_{23} = 5 + (-4) = 1$
$P_3 \longleftarrow 2 = \{0, 1, 2, 1\}$
fin si
fin para

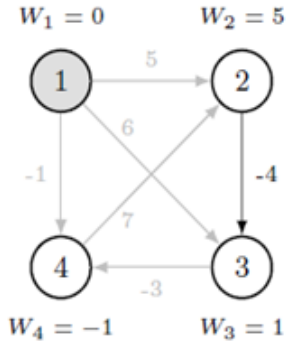


Figura 4.31: Grafo iteración 2

Iteración3

Para cada arco $(i, j) \in A$ hacer
Arco$(3, 4)$:
Si $W_j > W_i + d_{ij}$, entonces
$W_4 \longleftarrow W_3 + d_{34} = 1 + (-3) = -2$
$P_4 \longleftarrow 3 = \{0, 1, 2, 3\}$
fin si
fin para

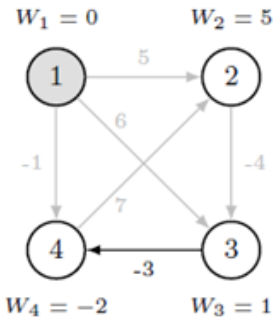


Figura 4.32: Grafo iteración 3

Iteración4

Para cada arco $(i, j) \in A$ hacer
Arco$(4, 2)$:
Si $W_j > W_i + d_{ij}$, entonces
fin si
fin para

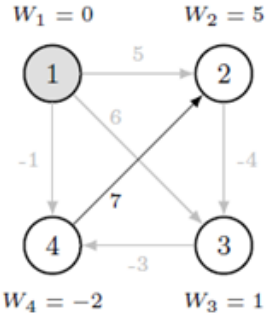


Figura 4.33: Grafo iteración 4

La siguiente etapa es la comprobación de ciclos negativos, lo cual se realiza de la forma que se expone a continuación:

Para cada arco $(i, j) \in A$ hacer
Arco$(4, 2)$:
Si $W_j > W_i + d_{ij}$, entonces
  no hay solución
fin si
fin para

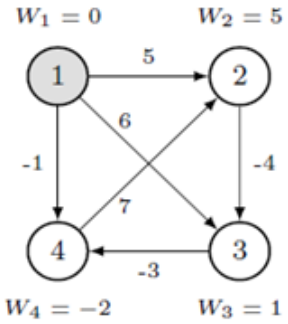


Figura 4.34: Grafo

Una vez desarrolladas cada una de las etapas anteriores, lo que sigue es la construcción del árbol o el diseño propiamente dicho de la ruta más corta.

Para cada arco $(i, j) \in A$ hacer
Arco$(4, 2)$:
Si $W_j > W_i + d_{ij}$, entonces
  no hay solución
fin si
fin para

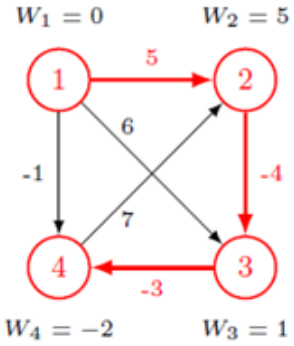


Figura 4.35: Ruta más corta

Por tanto la salida que genera el algoritmo de Bellman-Ford es la siguiente:

$V^T = \{1, 2, 3, 4\}$
$A^T = \{(1, 2), (2, 3), (3, 4)\}$
$W = \{0, 5, 1, -2\}$

Solución 2: (Implementando la herramienta computacional WinQsb)

A continuación resolveremos este problema de la ruta más corta mediante el uso de la herramienta computacional WinQsb. Recordemos que con esto, la idea central es la confirmación de los resultados encontrados por medio del algoritmo de Bellman-Ford:

a. Elegir módulo de modelado de red.

Figura 4.36: Módulo de Modelo de Redes del WinQsb

b. Seleccionar nuevo problema y Shortest Path Problem

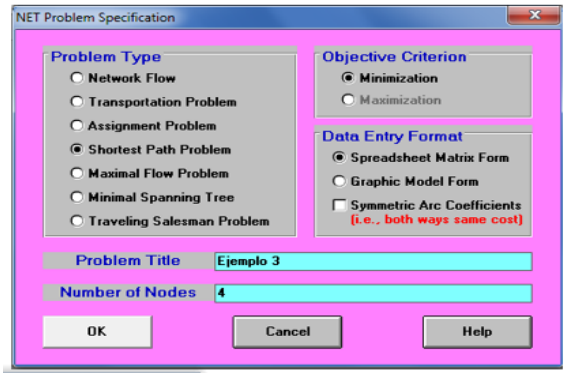


Figura 4.37: Especificaciones del ejemplo 4.3

c. Ingresar los datos del problema

From \ To	Nodo 1	Nodo 2	Nodo 3	Nodo 4
Nodo 1		5	6	-1
Nodo 2			-4	
Nodo 3				-3
Nodo 4		7		

Figura 4.38: Instancia del ejemplo 4.3

d. Mostrar la solución del problema

	From	To	Distance/Cost	Cumulative
1	Nodo 1	Nodo 4	-1	-1
	From Nodo 1	To Nodo 4	=	-1
	From Nodo 1	To Nodo 2	=	5
	From Nodo 1	To Nodo 3	=	1

Figura 4.39: Solución del ejemplo 4.3

4.4 Problemas propuestos

1. Supongamos que se envía energía eléctrica desde una planta (nodo 1) a una cierta ciudad (nodo 6) a través de una serie de subestaciones (nodo 2 al 5). Las distancias entre las combinaciones factibles se muestran en la figura. Determine la forma de enviar energía de modo de minimizar la distancia a recorrer.

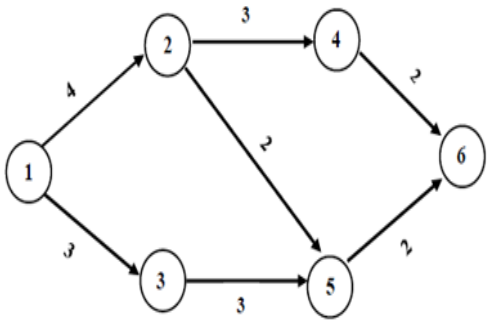


Figura 4.40: Grafo del problema 1

2. Una persona tiene que trasladarse a diario del pueblo 1 al pueblo 8, está estudiando cuál es el trayecto más corto usando un mapa de carreteras, el cual es representado por el siguiente grafo.

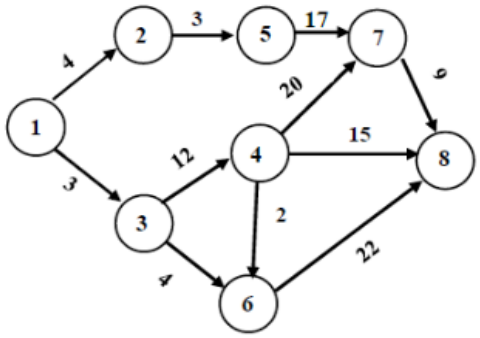


Figura 4.41: Grafo del problema 2

3. Una ciudad tiene siete subdivisiones. El alcalde desea instalar líneas telefónicas, para asegurar la comunicación entre todas las subdivisiones. En la figura se dan las distancias entre las subdivisiones. Determine la distancia mínima desde el nodo 1 a cada nodo del siguiente grafo.

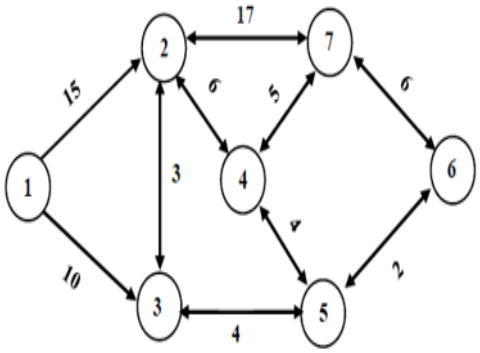


Figura 4.42: Grafo del problema 3

4. Una agencia de entregas de encomienda tiene una tarifa de cobro diferenciada por llevar un determinado material de un lugar a otro. En el grafo que se muestra a continuación se representan los diversos lugares y los correspondientes montos por traslado. Una persona que está ubicada en el sitio S desea enviar una encomienda al lugar T. Determinar porque trayectoria debe enviar la encomienda de tal manera que el pago por el monto sea mínimo.

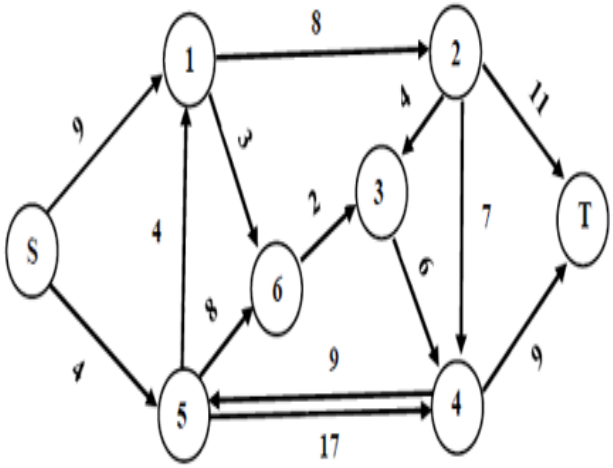


Figura 4.43: Grafo del problema 4

5. Para una empresa constructora, supóngase que el nodo 1 es el centro de almacenamiento y abastecimiento de la compañía. Con frecuencia se realizan viajes diarios desde el nodo 1 hasta los demás nodos o sitios de construcción. Utilizando el nodo 1 como nodo inicial, obtenga la ruta más corta desde este nodo hasta cada uno de los demás nodos de la red. A continuación se muestra el grafo que corresponde a la situación antes planteada.

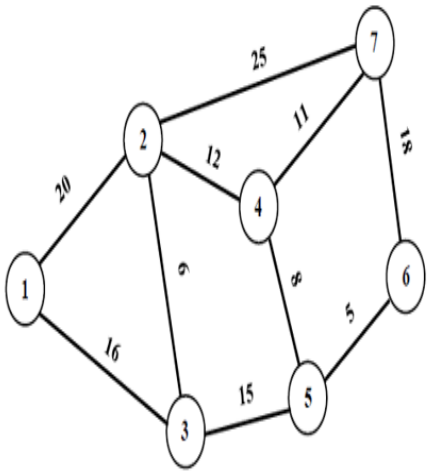


Figura 4.44: Grafo del problema 5

4.5 Problemas resueltos

Problema 1
Solución: (Aplicando el algoritmo de Dijkstra)

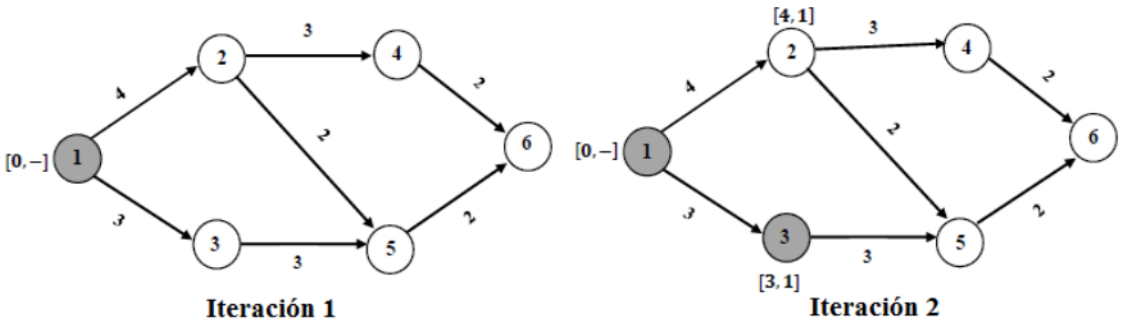


Figura 4.45: Iteración 1 y 2

Es posible apreciar que en la iteración 1 se etiqueta y fija el nodo 1, puesto que este nodo representa el sitio de partida. En la iteración 2 se etiquetan los nodos 2 y 3, de los cuales se fija el nodo 3 dado que la distancia de 1 a 3 es la mínima en ese momento.

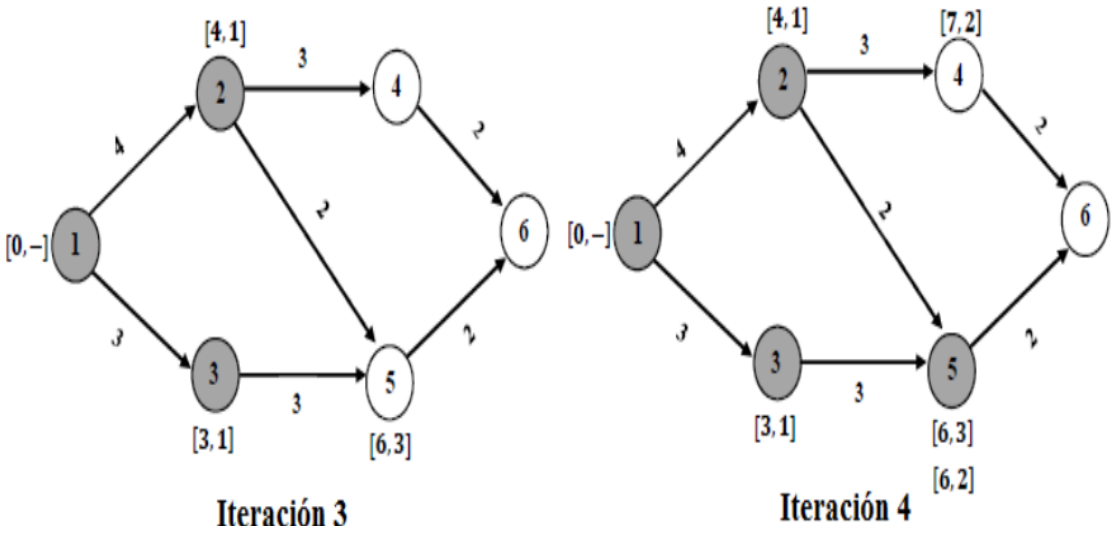


Figura 4.46: Iteración 3 y 4

En la iteración 3 etiquetamos al nodo 5 y se colorea, es decir se fija el nodo 2. Así mismo, en la interación 4, pasamos a etiquetar al nodo 5 y nodo 4 y el nodo de menor distancia acumulada, es decir fijamos al nodo 5.

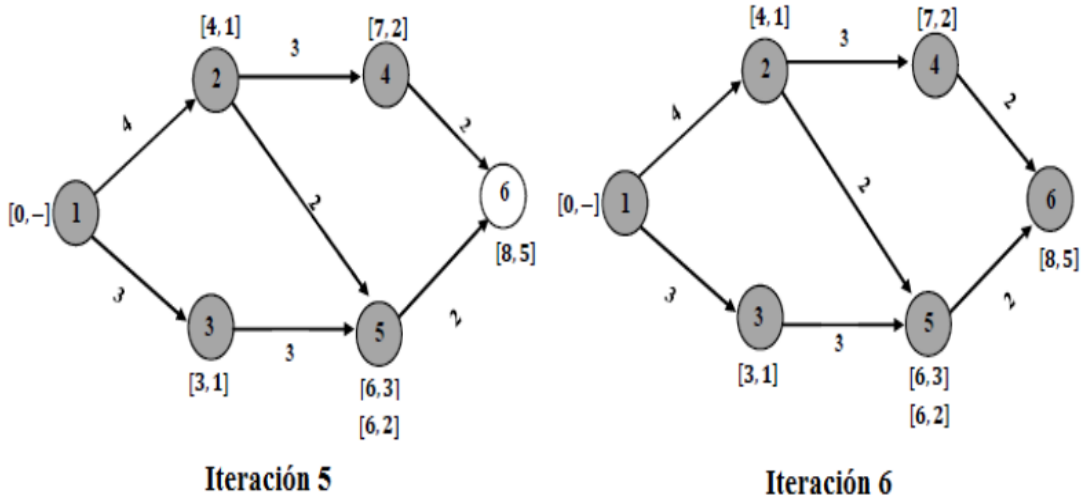


Figura 4.47: Iteración 4 y 5

En la iteración 5 etiquetamos al nodo 6 y luego coloreamos el nodo 4 dado que a dicho nodo se encuentra la menor distancia acumulada, de todos los nodos no fijados. Asi mismo, en la interación 6, pasamos colorear el nodo 6, que en este caso es la única opción.

La ruta más corta entre 1 y 6 es la siguiente:

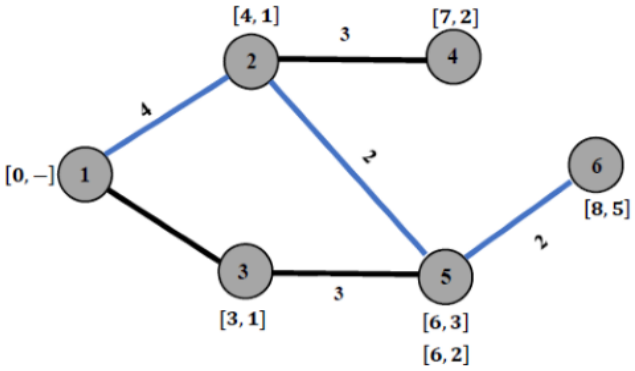


Figura 4.48: Ruta más corta del problema 1

Problema 2
Solución: (Aplicando el algoritmo de Dijkstra)

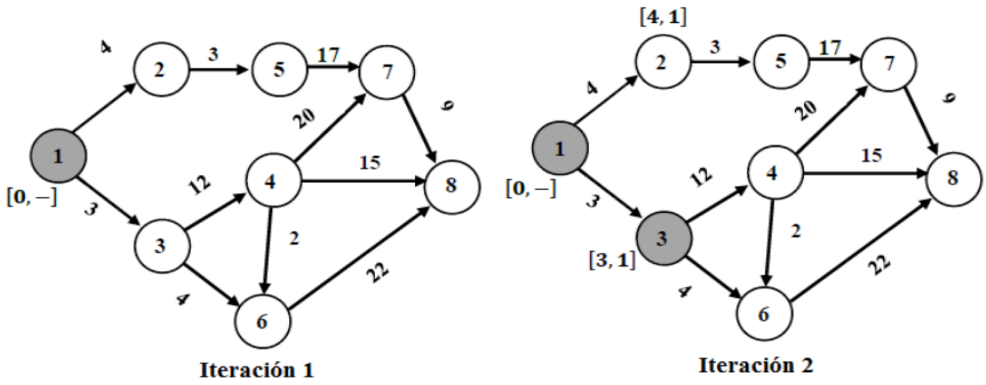


Figura 4.49: Iteraciones 1 y 2

Podemos ver que en la iteración 1 se etiqueta y fija el nodo 1 puesto que este nodo representa el sitio de partida. Y luego en la iteración 2 se etiquetan los nodos 2 y 3, de los cuales se fija el nodo 3 dado que la distancia de 1 a 3 es la mínima hasta ese momento.

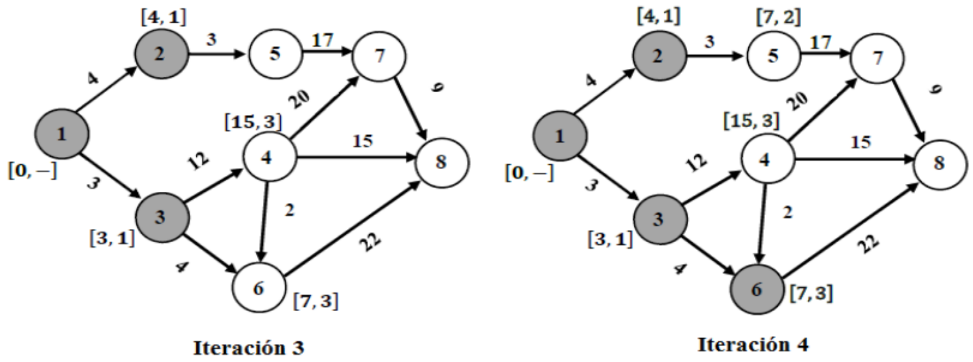


Figura 4.50: Iteraciones 3 y 4

En la iteración 3 etiquetamos al nodo 4 y nodo 6 y fijamos el nodo 2. De forma análoga, en la iteración 4, pasamos a etiquetar únicamente al nodo 5 y de los nodos etiquetados y sin fijar, fijamos al nodo que tiene menor distancia acumulada, siendo en este caso el nodo 6.

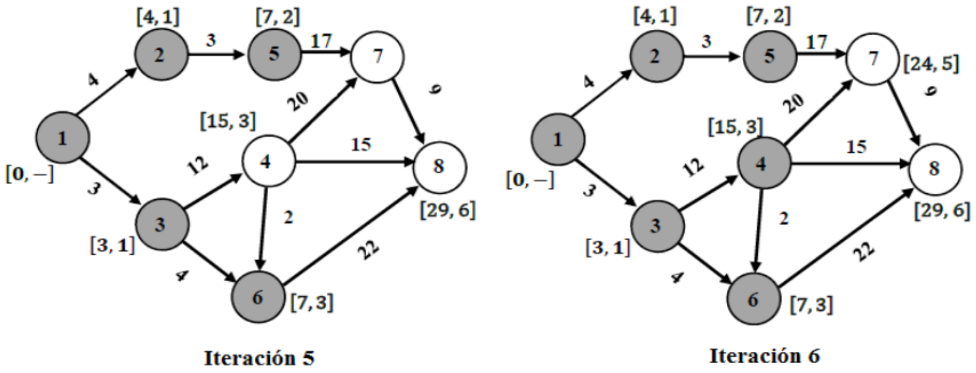


Figura 4.51: Iteraciones 5 y 6

En la iteración 5 etiquetamos al nodo 8 y de los nodos etiquetados y sin fijar, procedemos a fijar al nodo 5 dado que tiene la menor distancia acumulada. De igual manera, en la iteración 6, pasamos a etiquetar únicamente al nodo 7 y fijamos al nodo 4.

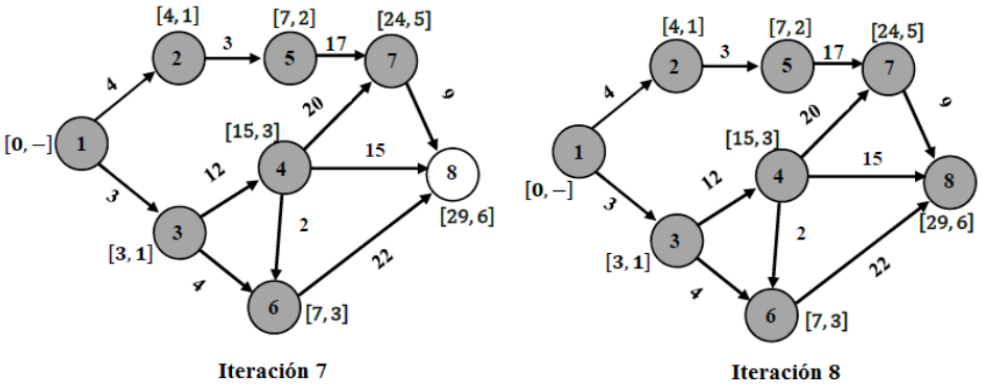


Figura 4.52: Iteraciones 7 y 8

En la iteración 7 procedemos a fijar al nodo 5 dado que tiene la menor distancia acumulada de los nodos etiquedos y sin fjiar. Luego, en la iteración 8, pasamos a dejar fijo al nodo 8.

La ruta más corta entre 1 y 6 es la siguiente:

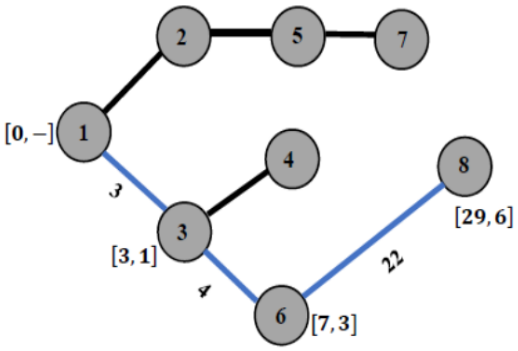


Figura 4.53: Ruta más corta del problema 2

Problema 3
Solución: (Aplicando el algoritmo de Dijkstra)

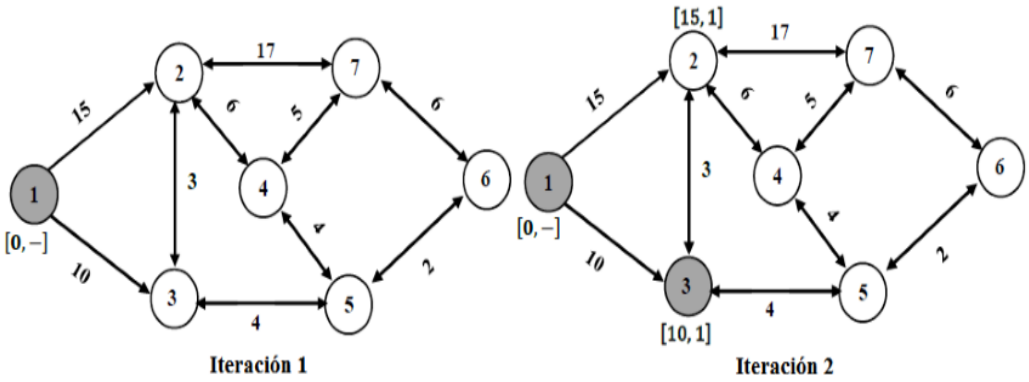


Figura 4.54: Iteración 1 y 2

En la iteración 1 se procede a etiquetar y fijar el nodo 1 ya que este nodo es el inicial. Seguidamente en la iteración 2 se etiquetan los nodos 2 y 3, de los cuales fijamos el nodo 3 dado que la distancia de 1 a 3 es la mínima hasta ese momento.

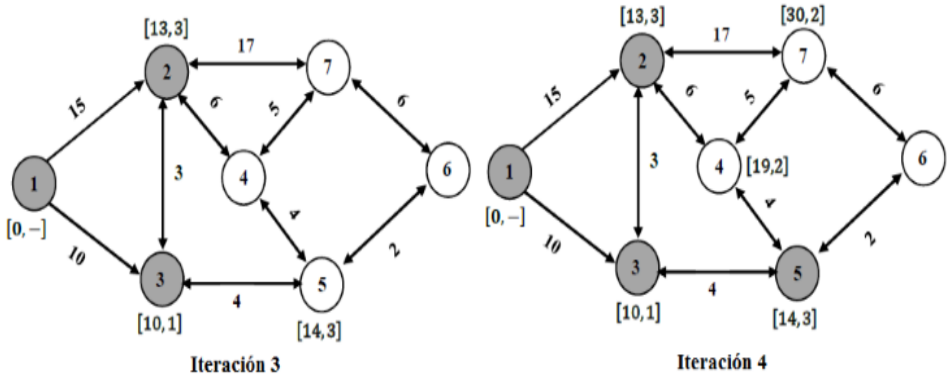


Figura 4.55: Iteración 3 y 4

En la iteración 3 etiquetamos al nodo 5, luego de los dos nodos etiquetados y sin fijar, procedemos a fijar al nodo 2 dado que tiene la menor distancia acumulada. De la misma manera, en la iteración 4, etiquetamos al nodo 7 y al nodo 4 y fijamos al nodo 5.

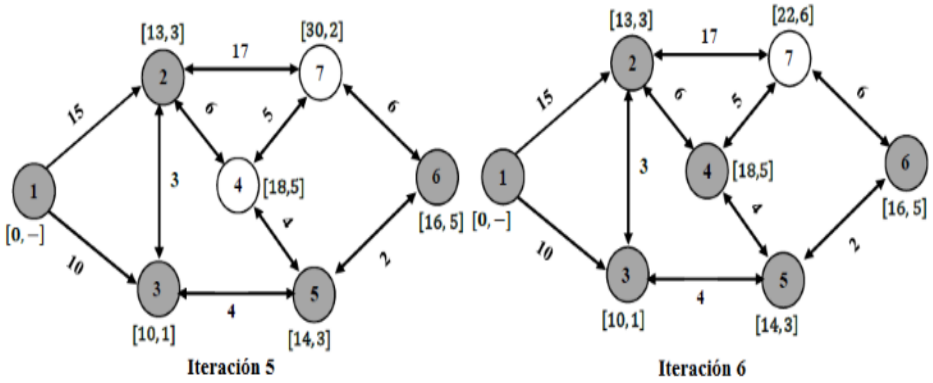


Figura 4.56: Iteración 5 y 6

Seguidamente, en la iteración 5 etiquetamos al nodo 6, luego de los dos nodos etiquetados y sin fijar, procedemos a fijar al nodo 6 ya que tiene la menor distancia acumulada. De igual forma, en la iteración 6, actualizamos la etiqueta del nodo 7 y fijamos al nodo 4.

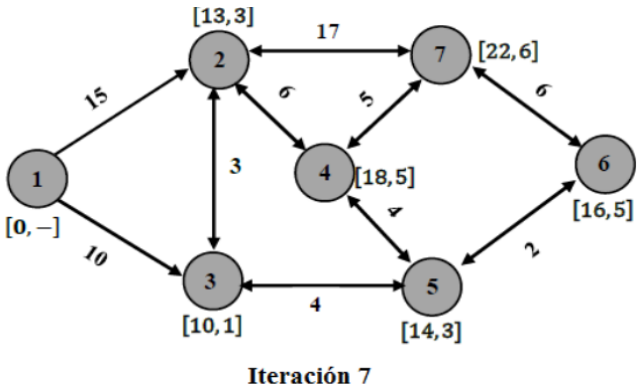


Figura 4.57: Iteración 7

En la iteración 7 solamente se procede a fijar el nodo 7 y con esto obtenemos la ruta más corta entre 1 y 6 es:

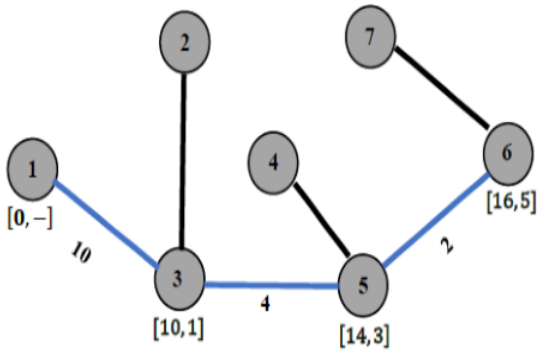


Figura 4.58: Ruta más corta del problema 3

Problema 4
Solución: (Aplicando el algoritmo de Dijkstra)

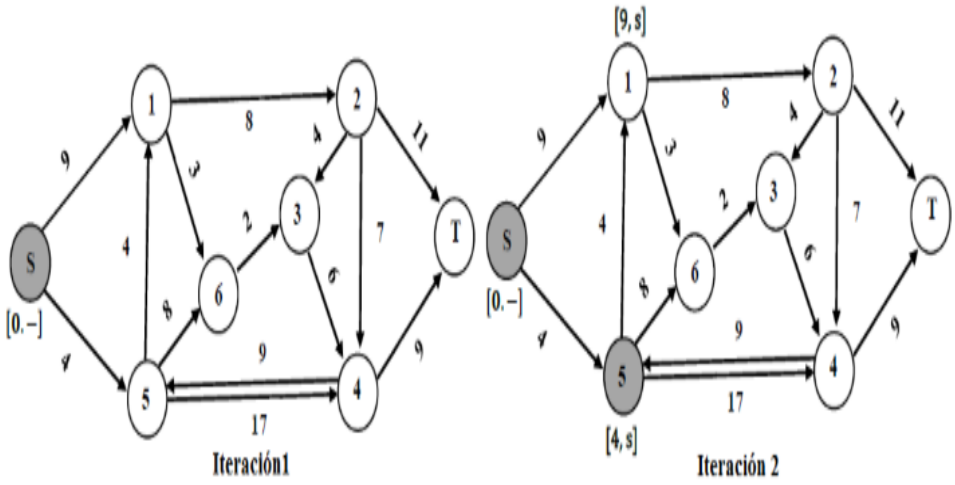


Figura 4.59: Iteración 1 y 2

En la iteración 1 etiquetamos y fijamos el nodo 1 ya que este nodo es el inicial. Luego en la iteración 2 se etiquetan los nodos 1 y 5, de los cuales fijamos el nodo 5 dado que la distancia de 1 a 3, hasta ese momento es la mínima.

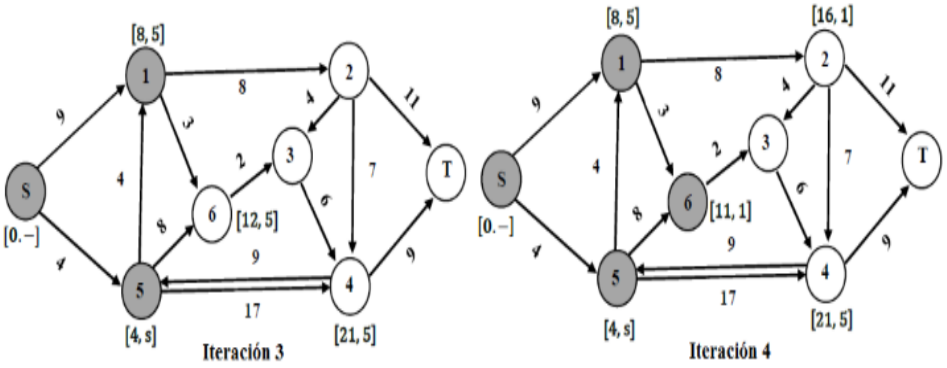


Figura 4.60: Iteración 3 y 4

Luego, en la iteración 3 etiquetamos al nodo 4 y 6, posteriormente, de los dos nodos etiquetados y sin fijar, procedemos a fijar al nodo 1 debido a que tiene la menor distancia acumulada. A su vez en la iteración 4, etiquetamos al nodo 2 y actualizamos la etiqueta del nodo 6 y fijamos al nodo 6.

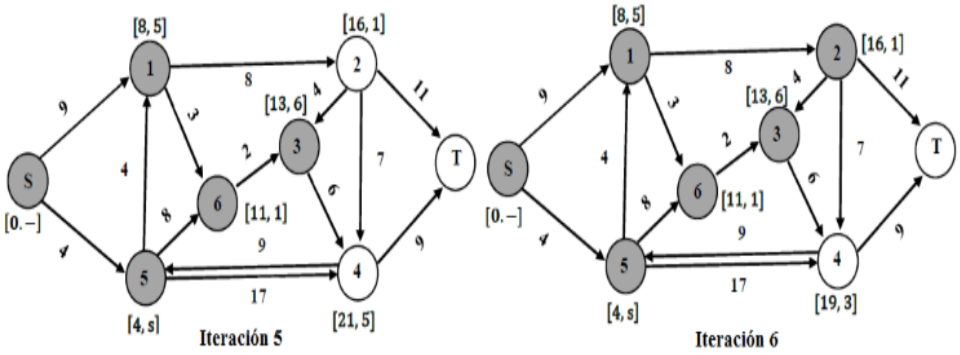


Figura 4.61: Iteración 5 y 6

Posteriormente, en la iteración 5 etiquetamos al nodo 3, luego usando el mismo criterio para fijar los nodos, procedemos a fijar al nodo 3. En cambio en la iteración 6, etiquetamos al nodo 2 y actualizamos la etiqueta del nodo 4 y fijamos al nodo 2.

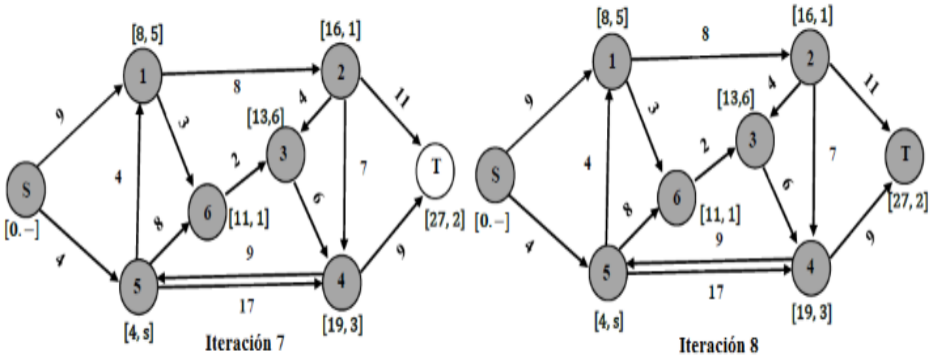


Figura 4.62: Iteración 7 y 8

En la iteración 7 fijamos al nodo 4 y en cambio en la iteración 8 fijamos al nodo T. Luego la ruta más corta es de S a T es:

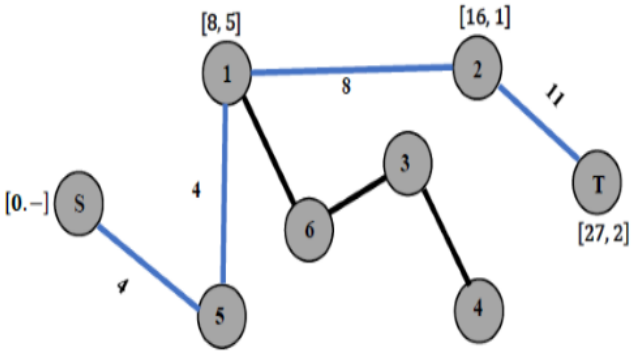


Figura 4.63: Ruta más corta del problema 4

Problema 5
Solución: (Aplicando el algoritmo de Dijkstra)

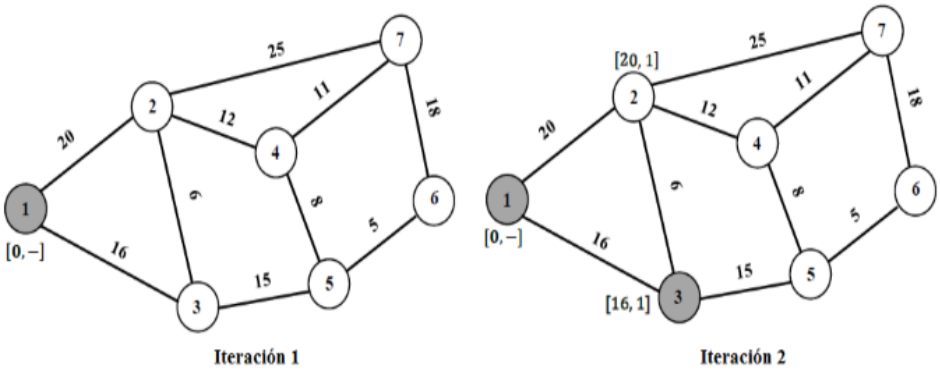


Figura 4.64: Iteración 1 y 2

Podemos apreciar que en la iteración 1 se etiqueta y fija el nodo 1, dado que es el nodo inicial. Luego, en la iteración 2 se etiquetan los nodos 2 y 3, de los cuales fijamos el nodo 3 ya que la distancia de 1 a 3 es la mínima haste ese momento.

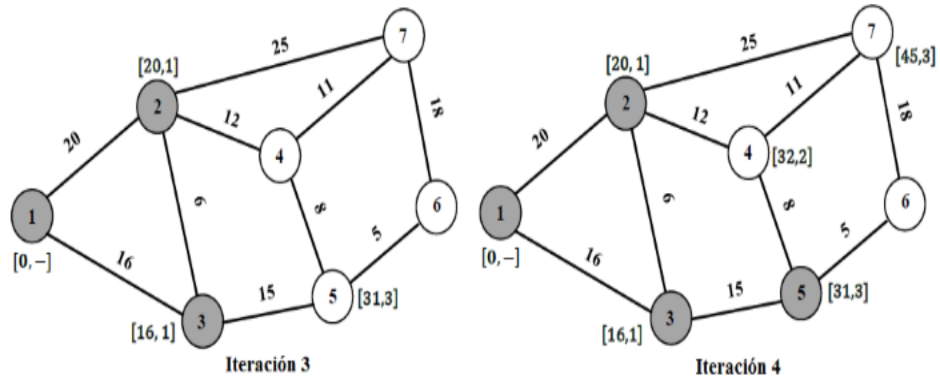


Figura 4.65: Iteración 3 y 4

Luego, en la iteración 3 etiquetamos al nodo 5, luego de los dos nodos etiquetados y sin fijar, procedemos a fijar al nodo 2 debido a que tiene la menor distancia acumulada. Después, en la iteración 4, etiquetamos al nodo 2 y 7, y fijamos al nodo 5.

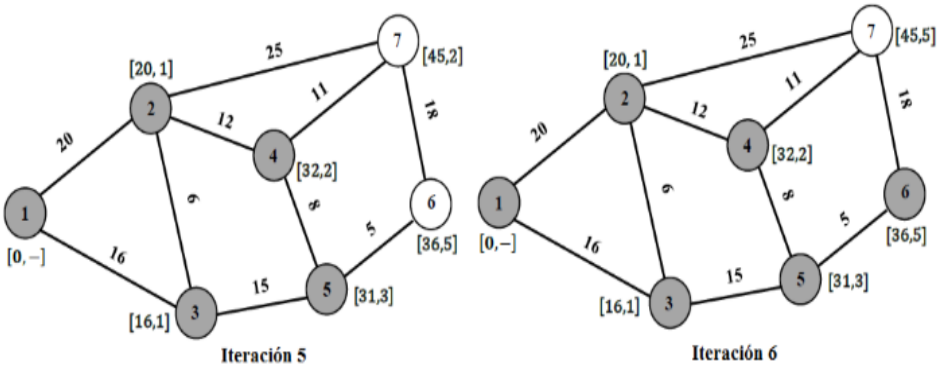


Figura 4.66: Iteración 5 y 6

Podemos ver que en la iteración 5 etiquetamos al nodo 6 y fijamos al nodo 4 debido a que tiene la menor distancia acumulada. Después, en la iteración 6, fijamos al nodo 6 .

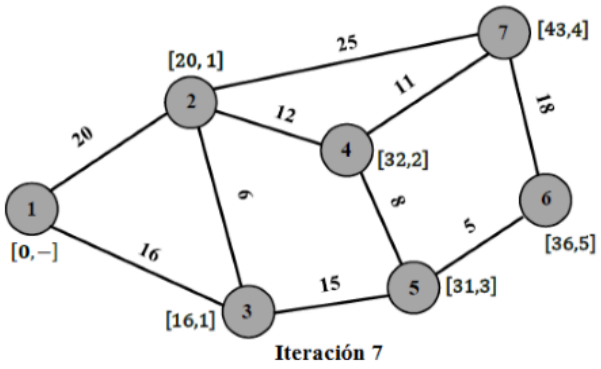


Figura 4.67: Iteración 7

En la última iterecaion, es decir la iteración 7, solamente procedemos a fijar el nodo 7. Y de esta manera, la ruta más corta de 1 a 7 es la siguiente:

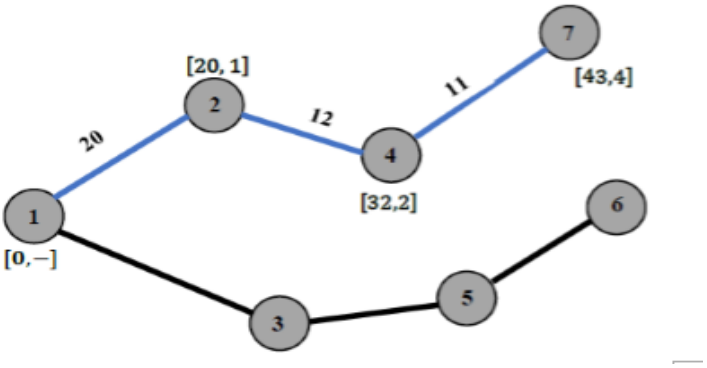


Figura 4.68: Ruta más corta del problema 5

4.6 Bibliografía

- Bazaraa, S., Javis, John J. (1996). *Programación lineal y flujo de redes*. México: Limusa. Quinta edición.
- Cormen, T., Leiserson, C., Rivest, R., Stein, C. (2001). *Introduction to algorithms (second edition)*. The MIT Press.
- Cherkassky, Boris V y otros. (1996). *Shortest paths algorithms: theory and experimental evaluation. Mathematical Programming*. Ser. A 73 (2): 129-174. Doi: 10.1016/0025-5610(95)00021-6. MR 1392160.
- Bellman, R. (1958). *On a routing problem. Quarterly of Applied Mathematics* 16: 87-90. MR 0102435.
- Dijkstra, E. W. (1959). *"A note on two problems in connexion with graphs"*. Numerische Mathematik 1: 269-271. doi:10.1007/BF01386390.
- Floyd, R. (1962). *Algorithm 97 (Shortest Path)*. Communications of the ACM, 5(6):345.
- Ford, L. R. (1956). *Network Flow Theory*. Santa Mónica, California: RAND Corporation.
- Hillier, F. y Lieberman, G., (2010). *Introducción a la investigación de Operaciones*. (9na. ed.). México: Ed. McGraw-Hill.
- Izar, J. (2008). *Investigación de Operaciones*. México: Editorial Trillas.
- Kenneth H. Rosen (2003). *Discrete Mathematics and Its Applications*, 5ª Edición. Addison Wesley.
- Meyer, U., Sanders, P. (2003). *A parallelizable shortest path algorithm. J. of Algorithms*. 49, 114-152.
- Moore, E. F. (1959). *The shortest path through a maze. Proceedings of an International Symposium on the Theory of Switching* (Cambridge, Massachusetts, 2-5 April 1957). Cambridge: Harvard University Press. pp. 285-292.
- Ríos, S. (1996). *Investigación operativa. Programación lineal y aplicaciones*. Madrid: Ed. Centro de Estudios Ramón Areces.
- Salazar, J. (2001). *Programación matemática*. Madrid: Ed. Díaz de Santos.
- Pinilla, V. (2005). *Investigación Operacional*. Universidad de Los Andes. Bogotá. 2005.

- Taha, H. (2012). *Investigación de operaciones*. (9na. ed.). México: Ed. Pearson.
- Thorup, Mikkel (1999). *Undirected single-source shortest paths with positive integer weights in linear time*. Journal of the ACM (JACM) 46 (3): 362-394.
- Warshall, S. (1962). *A theorem on boolean matrices*. Journal of the ACM, 9(1):11-12.

CAPÍTULO 5
El PROBLEMA DE CORTE

5.1 Introducción

El Problema de Corte, al igual que otros problemas de Optimización Combinatoria, es fácil de definir intuitivamente, sin embargo no es fácil de modelar formalmente y presenta un alto grado de dificultad lo cual hace difícil el manejo computacional. En este problema se tiene un conjunto de piezas de diferentes tamaños y formas que deben ser localizadas sobre un tablero de material de mayor tamaño sin superponerse unas sobre otras. El objetivo de tal disposición es maximizar el área utilizada de forma que se generen la menor cantidad de área desperdiciada.

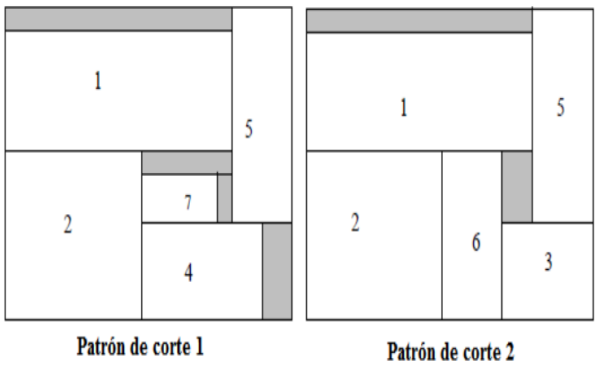


Figura 5.1: Patrones de corte en dos dimensiones

El Problema de Corte ha sido ampliamente estudiado en numerosas áreas de la industria y la investigación. Es un problema que se presenta en aquellas industrias en las que dentro de su proceso productivo es necesario cortar material que viene en dimensiones estándar, con el fin de obtener piezas en tamaños y formas requeridas, como por ejemplo: el corte de madera, de rollos de papel, de tela, de acero, entre otros. Para estas industrias es de gran importancia realizar este proceso de corte de una manera eficiente buscando minimizar el desperdicio y los demás costos asociados al proceso, teniendo en cuenta las restricciones técnicas y de demanda que el sistema en cuestión impone.

En el problema de Corte, los elementos pequeños son una lista de piezas demandadas, mientras que los objetos de mayor tamaño se conocen como stock o materia prima. Los clientes requieren las piezas, y las industrias o fábricas deben producir el conjunto de piezas demandadas usando la materia prima disponible para satisfacer las necesidades de sus clientes.

El Problema de Corte es un problema de gran complejidad tanto por las características y variables que incluye así como por las técnicas que se implementan para abordarlo, es un tema en permanente evolución y muchos investigadores han desarrollado diversos métodos para resolverlo. El interés en

dicho problema se basa en su aplicación práctica y por supuesto en el reto que constituye para la academia, ya que en general, es computacionalmente difícil de resolver. Por lo tanto, se evidencia la necesidad de explorar nuevos métodos de solución o la validación y ajuste de métodos y modelos existentes para problemas particulares en industrias específicas y sus respectivas necesidades, para así lograr la utilización eficiente de los recursos escasos en los procesos inherentes a la programación del corte y su inventario.

Este problema pertenece a la categoría de problemas de optimización combinatoria denominados NP-completos debido a que el espacio de soluciones crece de forma exponencial, dependiendo de las condiciones y restricciones que el problema involucre y de las diferentes combinaciones de corte que puedan existir. Si por ejemplo se tienen piezas a ser ubicadas, entonces el espacio de soluciones estará dado por $2^n \cdot n!$

El Problema de corte fue formulado por primera vez en 1939 por el economista ruso Kantorovich, y en 1951, junto con Zalgaller sugirieron resolverlo mediante técnicas de Programación Lineal buscando la utilización económica del material en la fase de corte. Posteriormente Gilmore y Gomory (1961), propusieron un modelo para la resolución de este tipo de problemas expresados en una sola dimensión, como por ejemplo, el largo o el ancho. Sus primeros intentos para resolverlo fueron a través de métodos analíticos; donde en un principio determinaron los patrones de corte posibles, que son descritos como el número de referencias de cada tipo que se cortan en longitudes estándar; posteriormente, la solución se logró utilizando un modelo matemático basado en estos patrones.

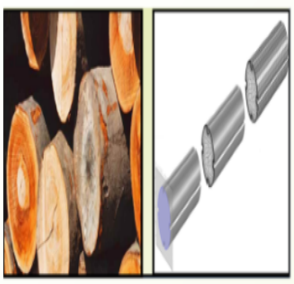


Figura 5.2: Corte en una dimensión

Ya para 1965, las investigaciones respecto al problema se extendieron a los problemas en dos dimensiones, los cuales son considerados como NP-hard. Estos problemas son abordados por Gilmore y Gomory como un problema unidimensional en dos etapas; y han presentado un enfoque de solución basado en una formulación de programación entera con una técnica de generación de columnas, donde cada columna representa un patrón de corte posible, y es generado por la solución del problema bidimensional de la mochila.

En 1977 Christofides y Withlock, propusieron un algoritmo exacto de bósqueda en árbol para resolverlo, utilizando para ello el algoritmo previamente propuesto por Gilmore y Gomory, quienes resuelven un problema de características similares. Por otro lado Wang (1983), propone un algoritmo de desarrollo incremental del patrón solución. Tal algoritmo es posteriormente mejorado en los trabajos de Vasko (1989) y Oliveira y Ferreira (1990).

Por otro lado, se han propuesto algoritmos heurísticos para la generación de patrones de corte factibles para producir las columnas de un problema de programación entera. Según Farley (1990) los patrones de corte pueden ser generados a través de la programación dinámica y por métodos mixtos. Obteniendo un límite para los problemas de dos dimensiones con un gran número de variables estructurales y la resolución del dual del problema de corte.

Por su parte, Yang y Weng (2006) plantean en su investigación un enfoque mejorado de Búsqueda Tabú para la resolución de problemas de corte unidimensional, el cual es tratado como un problema de secuencia e incorpora una función objetivo mixta haciendo uso del concepto de reutilización de materiales, permitiendo así elegir el plan de corte con el mínimo desperdicio, y lograr un mayor acercamiento hacia la obtención de una solución óptima.

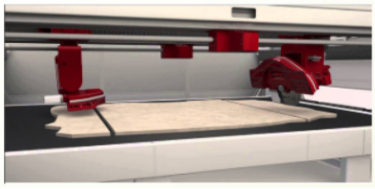


Figura 5.3: Corte en dos dimensiones

Así mismo, Parajón, Álvarez y otros (2007) desarrollan un Procedimiento de Búsqueda Adaptable Aleatorizado y Codicioso (GRASP) para la restricción bidimensional de dos etapas para el Problema de Corte. Este es un problema de corte especial en el que el corte se realiza en dos fases.

En la primera fase, el rectángulo de stock se corta por su anchura en diferentes tiras verticales y en la segundo fase, cada una de estas tiras se procesa para obtener las piezas finales. También se proponen dos algoritmos diferentes basados en GRASP. Uno es "orientado a la pieza", mientras que el otro es "orientado a la tira". Ambos procedimientos son rápidos y proporcionan soluciones de diferentes estructuras a este problema de corte.

Mobasher y Ekici (2012) proponen un modelo de Programación Lineal Entera Mixta que tiene como objetivo minimizar el costo total de producción considerando el costo de materia prima y los costos de alistamiento. Para resolver este problema proponen como método de solución dos algoritmos de búsqueda local y la técnica de generación de columnas basada en un algoritmo heurístico.

Lo antes expuesto es una muestra de cómo han surgido numerosas investigaciones que abordan diferentes problemas según el tipo de dimensión y desde diversos enfoques tales como los métodos exactos, heurísticos y meta heurísticos, pero aún no existe un método global establecido para dar solución

a este tipo de problemas, debido a la complejidad asociada. Otros enfoques utilizados han sido la relajación lagrangiana, métodos de eliminación sucesiva, programación dinámica, procedimientos heurísticos secuenciales, entre otros.

Figura 5.4: Corte en tres dimensiones

Existen diferentes criterios para clasificar los problemas de empaquetado, como son el tamaño y la cantidad de las piezas, las dimensiones del problema, etc. En estos problemas, la estrecha relación que guardan las características de los materiales y su aplicación a procesos específicos ha dado lugar a múltiples variantes. La primer tipología para permitir un sistema de notaciones y definiciones unificado y consistente, fue propuesta por Dickhoff (1990) y fue mejorada posteriormente por Wäscher, Haussner y Schumann (2007). Esta última tipología establece que los nuevos criterios para la definición de estos problemas son: 1) dimensionalidad (uno/dos/tres); 2) tipo de asignación (maximización/minimización); 3) surtido de objetos pequeños (idénticos/débilmente heterogéneos/fuertemente heterogéneos); 4) surtido de objetos grandes (uno/múltiples) y 5) forma de los pequeños ítems (rectángulos, círculos, cajas, cilindros).

Cada tipo de problema tiene por tanto una codificación de acuerdo a sus características. Como ejemplo ilustrativo, si se considera el clásico problema en el que se han de cortar barras grandes (objetos) disponibles en stock (todas ellas idénticas) en una de sus dimensiones con el fin de satisfacer las demandas a lo largo de un periodo de tiempo de barras más pequeñas (Ítems), el problema se denotaría como 1/V/I/R.

A partir de dos características (tipo de asignación y surtido de ítems pequeños) identifican cinco problemas básicos de Corte y Empaquetado:

- Problema de Empaquetado con Ítems Idénticos (Identical Item Parking Problem)
- Problema de Emplazamiento (Placement Problem)
- Problema de la Mochila (Knapsack Problem)
- Problema de Dimensión Abierta (Open Dimension Problem)
- Problema de Corte (Cutting Stock Problem)
- Problema de la Caja de Embalaje (Bin Packing Problem)

El Problema de Corte tiene diversas aplicaciones en las industria papelera, industria maderera, del vidrio, textil, de pieles y calzado, del metal, como también en el diseño de circuitos integrados, en el paginado de periódicos y sin lugar a dudas en la distribución ya que su aplicación permite obtener patrones de ubicación de productos que maximiza espacio de camiones de diferentes tipos. A continuación se expone una breve descripción de dichas aplicaciones:

- Industria papelera. En lo que concierne a la industria papelera se buscaba encontrar la manera óptima en que se puede cortar una bobina en pedazos más pequeños, también se intenta resolver el problema de asignación del orden en el caso de múltiples máquinas de corte en paralelo.
- Industria maderera. En la industria de corte de madera se aborda un problema similar al de la industria del papel, salvo que en este caso el corte en pequeñas piezas rectangulares se realiza mediante sierras circulares, aquí también se aborda el problema del equilibrio entre la producción con poco desperdicio y la producción con diseños más sencillos que permitan cortes con menos complicaciones.
- Industria del vidrio. El corte de vidrio es un proceso en el que en primer lugar el vidrio debe ser marcado y a continuación se corta sobre un eje plano.
- Industria textil. La industria textil también es un buen banco de pruebas en el que aplicar investigaciones sobre el empaquetado automatizado. El corte de prendas de vestir a partir de bobinas de tela, ofrece muchas dificultades y matices. Estos problemas pueden incluir dibujos en las bobinas y, por tanto, limitaciones en la rotación de las piezas por las orientaciones en las que se puede cortar la ropa. Además, a menudo se trata de formas muy irregulares y muchas veces el material presenta zonas defectuosas.
- Industria de pieles y calzado. La industria de corte de piel también ha sido objeto de algunos de los últimos trabajos de corte. Aquí se trata el problema del diseño de cortes, ya que el cuero posee una considerable variabilidad en su resistencia y en la calidad de diferentes áreas. Estos factores suelen restringir las posibles ubicaciones de las piezas. Por otro lado la variabilidad en los tipos de pies obligan a la supervisión directa de un operario bien durante todo el proceso o bien para identificar las zonas resistentes.
- El corte metálico contiene un gran número de aplicaciones reales. Por ejemplo, los costos de material pueden variar considerablemente, desde el corte de hojas finas de aluminio hasta perfiles de metal que pueden tener espesores de varios centímetros. Por lo general, esto tiene un impacto en la forma de abordar el problema, así por ejemplo el caso del aluminio la importancia de generar un diseño óptimo es probablemente secundaria frente a la producción de una solución rápida que permita el ahorro de costos de producción y mantenga la producción en funcionamiento. Sin embargo, en el caso de metales de gran espesor el costo por unidad de

material es alto y por tanto la optimización del corte supone un ahorro significativo en costes. Los casos del corte metálico es enorme; corte unidimensional, corte bidimensional; piezas regulares rectangulares, piezas circulares; corte con perfiles o bobinas iguales; corte con perfiles diferentes y stock limitado.

5.2 Formulación matemática del Problema de Corte

a. **Problema de Corte en una dimensión**

Dado un stock de barras de tamaño L (objetos grandes) que deben ser cortadas en piezas de tamaño l_i (pequeños ítems) para atender la demanda N_i, $i = 1, ..., m$ de las piezas de l_i. Las demandas son atendidas decidiendo sobre los distintos patrones (modelos de cortes) de la barra de tamaño L. El objetivo es minimizar el número de los grandes objetos utilizados.

El j-ésimo patrón o modelo de corte es una manera de dividir la barra de tamaño L en piezas de tamaño l_i y x_j es el número de veces que se utiliza el j-ésimo patrón de corte.

En la formulación del Problema de Corte de una dimensión como un programa no entero, Gilmore y Gomory (1960), la matriz A del problema lineal tiene m filas y un gran número de columnas, uno por cada posible patrón de corte. Así cada vector $(a_1, a_2, ..., a_m)$ de enteros no negativos satisfaciendo $l_1a_1 + l_2a_2 + \cdots + l_ma_m \leq L$ es una columna de la matriz.

Así, tenemos que el modelo matemático para el problema de corte unidimensional es:

$$\begin{aligned} \text{Minimizar} \quad & \sum_j x_j \\ \text{s.a} \quad & \sum_j a_{ij}x_j \geq N_i, \;\; i = 1, 2, ..., n \\ & x_j \geq 0 \end{aligned}$$

Y en forma matricial

$$\begin{aligned} \text{Minimizar} \quad & 1x \\ \text{s.a} \quad & Ax \geq N \\ & x \geq 0 \end{aligned}$$

Donde
$x = (x_1, x_2, ..., x_j)$ es un vector columna de las variables x_j y para cada columna de la matriz

A, donde x_j es el número de veces que el patrón j se utiliza.
$1 = (1, 1..., 1)$ es un vector fila de elementos todos iguales a uno.
$N = (N_1, ..., N_m)$ es un vector columna de las demandas N_i.
A : Matriz de m filas, cuyas columnas de estructura $(a_1, a_2, ..., a_m)$ son los posibles patrones de corte.
a_i: es el número de veces que la pieza $l_m(i = 1, ..., m)$ aparece en el patrón de corte.

Considerando x_j y N_i enteros, entonces estaríamos ante un problema de programación lineal entera. En la práctica se resuelve el problema de programación lineal asociado y redondeando la solución de este, podemos tener una solución satisfactoria para el problema de programación entera.

b. **Problema de Corte en dos dimensiones**

Dado un stock de láminas de dimensiones $L \times A$, deben ser cortadas en piezas de tamaño $l_i \times a_i$ proporcionando N_i piezas. Las N_i piezas es la demanda que debe ser atendida por un número cualquiera de láminas.

El corte se ejecuta del siguiente modo: Se selecciona un cierto número de patrones de corte rectangular, donde cada patrón (modelo) es definido como la manera de ajustar rectángulos menores $l_i \times a_i$ dentro del rectángulo mayor $L \times A$. El j-ésimo patrón de corte describe cómo una lámina deberá ser cortada para producir láminas menores de acuerdo a la demanda, y donde x_j es la cantidad de veces que el j-ésimo patrón de corte se utiliza. El objetivo del problema es atender las demandas usando el menor número de láminas.

El problema general de corte en dos dimensiones es formulado de forma semejante al de cortes en una dimensión:

$$\begin{array}{ll} \text{Minimizar} & 1x \\ \text{s.a} & Ax \geq N \\ & x \geq 0 \end{array}$$

Donde
$x = (x$»$1, x_2, ..., x_j)$ es un vector columna de las variables x_j y para cada columna de la matriz A, donde x_j es el número de veces que el patrón j-ésimo patrón de corte es utilizado.
$1 = (1, 1, ..., 1)$ es un vector fila de elementos todos iguales a uno.
$N = (N_1, ..., N_m)$ es un vector columna de las demandas.
A: Matriz de m filas, cuyas columnas de estructura $(a_1, a_2, ..., a_m)$ son los posibles modelos de

corte rectangular a cortar de $L \times A$.
a_i: es el número de rectángulos $l_m \times a_i (i = 1, ..., m)$ que ocurre en el modelo.

El problema de corte algunas veces es subdividido por el objetivo que persigue en dos subproblemas:

- Problema de selección o agrupamiento (The Assortment Problem) que tiene por objetivo determinar un subconjunto de láminas (barras) de las disponibles para satisfacer la demanda de piezas, cuya formulación es la que aparece en PC2D.
- Problema del desperdicio por corte (The Trim-Loss problem) que tiene por objetivo determinar un patrón de corte adecuado para cumplir con la demanda a partir del stock de láminas (barras) minimizando las pérdidas o desperdicio. Se conoce como problema de corte restricto (constrained stock cutting problem) cuando el número de piezas de un determinado tipo solicitado es limitado superiormente para todo patrón de corte; en caso de no existir un límite superior del número de piezas producidos, el problema se conoce como problema de corte no restricto (unconstrained stock Cutting problem).

No siempre es fácil clasificar los problemas de corte por su dimensión, así por ejemplo, problemas de carga en palletes espacialmente es un problema $3D$, sin embargo usualmente se considera como $2D$, pues la altura no se considera, pero si la colocación de los ítems es en camadas entonces se debe considerar una tercera dimensión. Similarmente, los contenedores son a menudo cargados construyendo primero pilas verticales y luego localizando las pilas horizontales en la base del contenedor. En ambos casos, se puede hablar de dimensionalidad "2 + 1" en vez de 3. De igual manera, cuando las planchas de vidrio son cortadas solamente por guillotina puede caracterizarse el Problema de Corte como de "dimensión 1+1" y no de 2 dimensiones.

c. **Problema de Corte en tres dimensiones**

En este caso se consideran ahora cada varilla como una caja B_j de tamaño fijo T_0; cada pieza a ser empaquetada tiene tamaño $t_i, (i = 1, ..., m)$. El objetivo es buscar una asignación (empaquetado) de todas las piezas $T = \{t_1, t_2, ..., t_m\}$ utilizando el menor número posible de cajas, tal que ninguna caja B_j (Bin) sea llenada sobrepasando su capacidad T_0 , así como ninguna pieza sea fraccionada en piezas menores (Bin Packing).

La formulación del problema de empaquetado de una dimensión como un problema de Programación Entera $0-1$ es la siguiente:

$$\begin{aligned}
\text{Minimizar} \quad & x_0 = \sum_{j=1}^{n} y_j \\
\text{s.a} \quad & \sum_{i=1}^{m} t_i x_{ij} \leq T_0 y_j, \quad j = 1, 2, ..., n \\
& \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, ..., m \\
& y_j \in \{0, 1\}, \quad j = 1, 2, ..., n \\
& x_{ij} \in \{0, 1\}, \quad \forall i, j
\end{aligned}$$

Donde

$$y_j = \begin{cases} 1 & \text{si } B_j \; es \; utilizada \\ 0 & en \; caso \; contrario \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{si } el \; item \; i \; es \; empaquetado \; en \; B_j \\ 0 & en \; caso \; contrario \end{cases}$$

La primera restricción nos dice que las piezas empaquetadas no sobrepasan la capacidad de la caja y la segunda restricción nos asegura que cada pieza es empaquetada solamente en una caja.

5.3 Tratamiento metodológico

Para el caso del Problema de Corte, el tratamiento metodológico consistirá en resolver problemas que posen un instancia de baja dimensión, para esto se implementará diferentes métodos de solución. Inicialemente se utilizará la técnica de Programación Lineal Entera, y para la cual usaremos el algoritmo Branch and Bound, auxiliándonos del WinQsb. Posteriormente se aborda el problema de cortar un tablero (bidimensional) en piezas más pequeñas, de tal manera que se optimice el beneficio. Para resolver dicho problema se implementará un algoritmo heurístico.

Ejemplo 5.1 *Supóngase que se tiene varillas de acero, en cantidad suficiente, de 12 metros de longitud cada una, y que se requieren cortes de las siguientes dimensiones y cantidades:*

Ciudad	Cantidad
3	100
5	150
7	200
8	85

Tabla 5.1: *Datos del ejemplo* 5.1

Se desea saber cómo cortar las varillas de 12 metros, de tal forma que el desperdicio sea mínimo. Se considera, en este caso, el desperdicio como cortes menores que $3m$ de longitud.

Solución: (Aplicando Programación Lineal Entera con WinQsb)

Podemos deducir que existen diversas formas posibles y lógicas de cortar cada varilla de $12m$, de tal forma que se obtengan las varillas de las dimensiones deseadas. Para poder formular el modelo deben describirse exhaustivamente todas estas formas. Para este caso, las combinaciones posibles o los patrones de corte factibles son los siguientes:

	Patrón de corte					
Dimensión de la varilla	**1**	**2**	**3**	**4**	**5**	**6**
3	4	2	1	1	0	0
5	0	1	0	0	2	1
7	0	0	1	0	0	1
8	0	0	0	1	0	0
Desperdicio(M)	0	1	2	1	2	0

Tabla 5.2: *Posibles cortes en el ejemplo* 5.1

Es decir que existen 6 formas lógicas de cortar las varillas de $12m$. Por ejemplo, el patrón de corte número 3 se obtiene una varilla de $3m$ y una varilla de $7m$, produciéndose un desperdicio de $2m$. Los demás patrones de corte de interpretan de forma análoga.

Con base a los 6 patrones de corte identificados en la solución 1, podemos definir las variables de decisión, de la siguiente manera:

Sea x_i el número de varillas de $12m$ a cortar según el patrón $i = 1, 2, ..., 6$

De manera que el modelo de Programación Lineal Entera será el siguiente:

$$\begin{array}{ll} \text{Minimizar} & D = x_2 + 2x_3 + x_4 + 2x_5 \\ \text{s.a} & 4x_1 + 2x_2 + x_3 + x_4 \geq 100 \\ & x_2 + 2x_5 + x_6 \geq 150 \\ & x_3 + x_6 \geq 200 \\ & x_4 \geq 85 \\ & x_i \geq 0, i = 1, 2, 3, 4, 5, 6 \end{array}$$

Una vez construido el modelo procederemos a resolverlo mediante Branch and Bound.

a. Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el WinQsb

Figura 5.5: Selección del problema

b. Ingresamos el modelo relajado

Variable -->	X1	X2	X3	X4	X5	X6	Direction	R. H. S.
Minimize	0	1	2	1	2	0		
C1	4	2	1	1	0	0	>=	100
C2	0	1	0	0	2	1	>=	150
C3	0	0	1	0	0	1	>=	200
C4	0	0	0	1	0	0	>=	85
LowerBound	0	0	0	0	0	0		
UpperBound	M	M	M	M	M	M		
VariableType	ontinuou	ontinuou	ontinuou	ontinuou	ontinuou	ontinuou		

Figura 5.6: Datos del modelo relajado

c. Obtenemos la solución:

	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
1	X1	3.7500	0	0	0	basic	0	2.0000
2	X2	0	1.0000	0	1.0000	at bound	0	M
3	X3	0	2.0000	0	2.0000	at bound	0	M
4	X4	85.0000	1.0000	85.0000	0	basic	0	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	2.0000
	Objective	Function	(Min.) =	85.0000	(Note:	Alternate	Solution	Exists!!)

Figura 5.7: Solución del modelo relajado

Por tanto obtenemos nuestra primera solución al problema $(3.75, 0, 0, 85, 0, 200)$ y $Z = 85$. Puesto que las soluciones no son enteras procedemos a ramificar tomando las variables cuyo valor no es entero, en este caso x_1.

Luego escogemos la variable x_1 y agregamos una rama en donde $x_1 \leq 3$, es decir la parte entera de 3.75

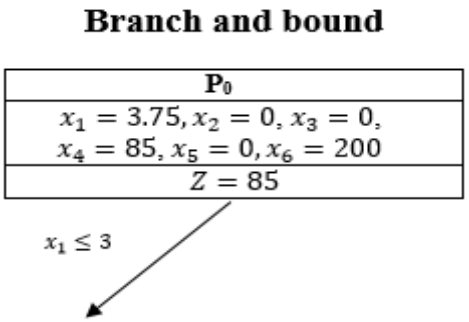


Figura 5.8: Solución P_0 ejemplo 5.1

Por lo tanto nuestro problema ahora queda de la siguiente forma:

$$\begin{aligned} \text{Minimizar} \quad & D = x_2 + 2x_3 + x_4 + 2x_5 \\ \text{s.a} \quad & 4x_1 + 2x_2 + x_3 + x_4 \geq 100 \\ & x_2 + 2x_5 + x_6 \geq 150 \\ & x_3 + x_6 \geq 200 \\ & x_4 \geq 85 \\ & x_i \geq 0, i = 1, 2, 3, 4, 5, 6 \end{aligned}$$

Resolviendo este problema mediante el WinQsb, tenemos:

	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
1	X1	3.0000	0	0	0	basic	-M	2.0000
2	X2	1.5000	1.0000	1.5000	0	basic	0	2.0000
3	X3	0	2.0000	0	1.5000	at bound	0.5000	M
4	X4	85.0000	1.0000	85.0000	0	basic	0.5000	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	1.5000
	Objective	Function	(Min.) =	86.5000	(Note:	Alternate	Solution	Exists!!)

Figura 5.9: Solución del modelo relajado

Es decir que la solución para dicho modelo es $(3, 1.5, 0, 85, 0, 200)$ y $Z = 86.5$. Luego procedemos a generar la rama que corresponde:

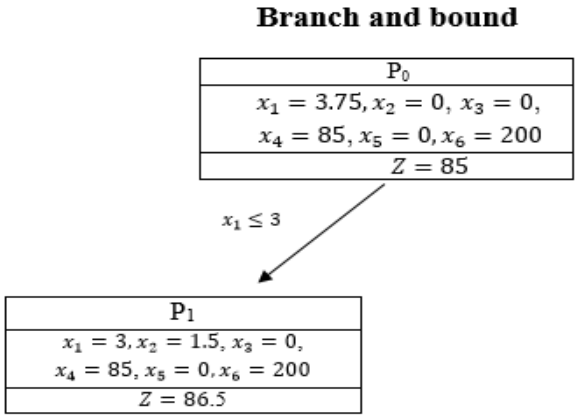


Figura 5.10: Solución P_0, P_1 ejemplo 5.1

Como se puede apreciar, tenemos una solución con variables no enteras y que además generan un valor de la función objetivo superior a la solución del problema inicial. Posteriormente, procedemos crear una rama al lado derecho y agregamos la restricción $x_1 \geq 4$. Y en este caso obtenemos el problema inicial más esta última restricción:

$$
\begin{aligned}
\text{Minimizar} \quad & D = x_2 + 2x_3 + x_4 + 2x_5 \\
\text{s.a} \quad & 4x_1 + 2x_2 + x_3 + x_4 \geq 100 \\
& x_2 + 2x_5 + x_6 \geq 150 \\
& x_3 + x_6 \geq 200 \\
& x_4 \geq 85 \\
& x_1 \geq 4 \\
& x_i \geq 0, i = 1, 2, 3, 4, 5, 6
\end{aligned}
$$

Seguidamente resolvemos el modelo y tenemos la siguiente tabla con la solución:

	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
1	X1	4.0000	0	0	0	basic	0	M
2	X2	0	1.0000	0	1.0000	at bound	0	M
3	X3	0	2.0000	0	2.0000	at bound	0	M
4	X4	85.0000	1.0000	85.0000	0	basic	0	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	2.0000
	Objective	Function	(Min.) =	85.0000	(Note:	Alternate	Solution	Exists!!)

Figura 5.11: Solución del modelo relajado

En este caso obtenemos como solución $(4, 0, 0, 85, 0, 200)$ y $Z = 85$. Creando la rama que corresponde tendríamos la gráfica:

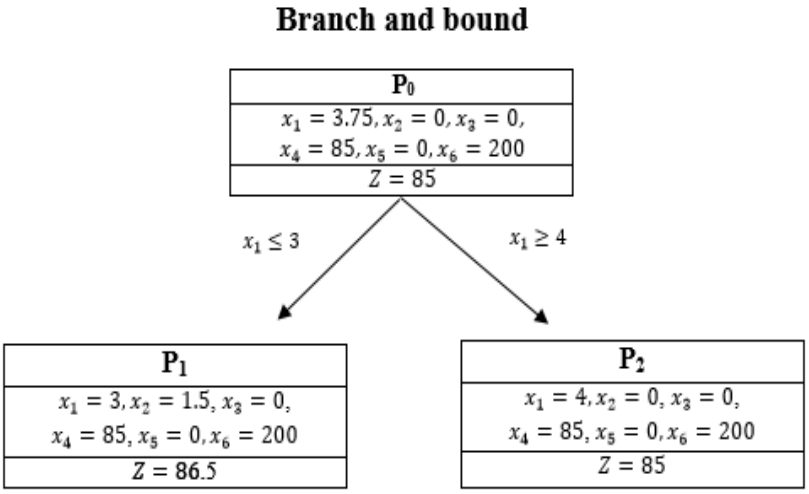


Figura 5.12: Solución P_0, P_1, P_2

En la rama del lado derecho nos encontramos con una solución entera, puesto que el valor de las variables de decisión es un número entero, y dado que en la rama izquierda nos encontramos con una solución no entera y que el valor de la función objetivo es superior a la que encontramos en el lado derecho, entonces podemos decir que ya hemos encontrado la solución óptima del problema $(4, 0, 0, 85, 0, 200)$ y con ella tenemos un valor óptimo de 85

Ejemplo 5.2 *Dado un tablero cuyas dimensiones son* $15m$ *de largo y* $13m$ *de ancho. Se desea cortar en piezas más pequeñas con dimensiones* $3m \times 7m$, $4m \times 5m$, $8m \times 4m$. *La idea es determinar los tipos y cantidad de cortes que se deben hacer para optimizar los beneficios. En la siguiente figura se muestra el tablero por cortar y los tipos de piezas demandados.*

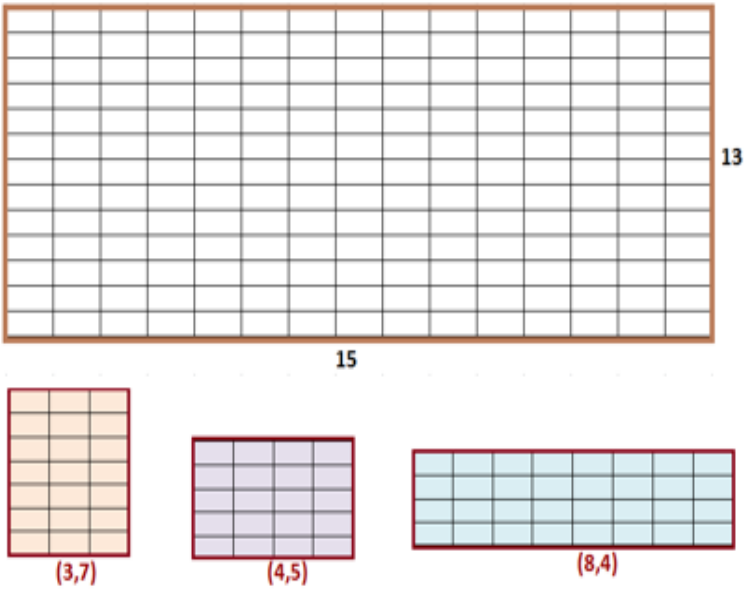


Figura 5.13: Tablero y piezas por cortar

Además, en la siguiente tabla se resume información referida a las piezas demandadas:

$\mathbf{i}$	$(\mathbf{l_i}, \mathbf{w_i})$	$\mathbf{v_i}$	$\mathbf{s_i}$	$\mathbf{r_i = \frac{v_i}{s_i}}$	$\mathbf{d_i}$	$[\frac{\mathbf{L}}{\mathbf{l_i}}][\frac{\mathbf{W}}{\mathbf{w_i}}]$
1	$(3, 7)$	78	21	3.72	6	$[\frac{15}{3}][\frac{13}{3}] = 5$
2	$(8, 4)$	65	32	2.03	8	$[\frac{15}{8}][\frac{13}{4}] = 2$
3	$(4, 5)$	28	20	1.40	10	$[\frac{15}{4}][\frac{13}{75}] = 6$

Tabla 5.3: *Información del ejemplo* 5.2

Solución: Para la cota superior BK1 el modelo es el siguiente:

BK1
$Bk_1(R) =$ Maximizar $\sum_{i \in S^*} v_i x_i$ donde $S^* = \{i / l_i \leq L, w_i \leq W\}$
s.a $\sum_{i \in S^*} s_i x_i \leq LW$
$0 \leq x_i \leq min\{d_i - n_i, [L/l_i][W/w_i]\}, \quad i = 1, 2, ..., m$

Tabla 5.4: *Algoritmo Corte*

A continuación se muestran cada una de las etapas del algoritmo:

Etapa 0

$R = \{(15, 13)\}$ Rectángulo original

$P = \{\} = \phi$ Conjunto de piezas cortadas

$V_T = 0$ Valor total de piezas cortadas

$S = \{(l_i, w_i) : i \in I\} = \{(3, 7), (8, 4), (4, 5)\}$ Piezas que se deben cortar

$n_1 = n_2 = n_3 = 0$ Número de piezas cortadas de cada tipo

Etapa 1

$R_k = \{(15, 13)\}$

$\{(3, 7), (8, 4), (4, 5)\}$ Piezas que se pueden cortar en R_k

Iniciamos el procedimiento para la Pieza $(3, 7)$

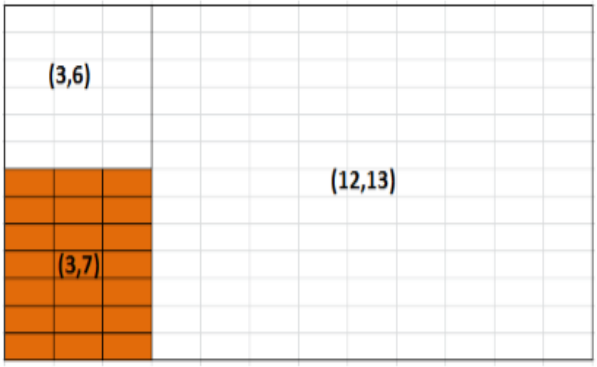


Figura 5.14: Tablero y piezas por cortar

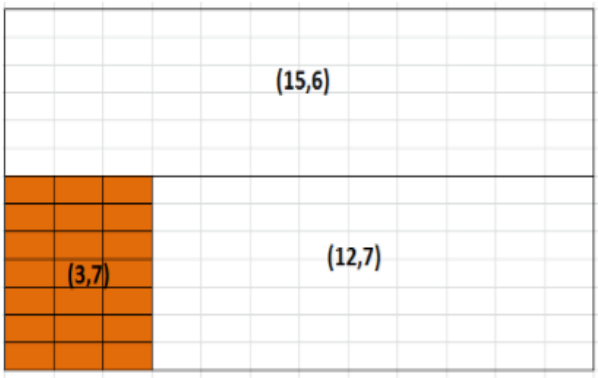


Figura 5.15: Tablero y piezas por cortar

Ahora procedemos a calcular los estimadores de los 4 tableros resultantes, es decir:

$e_1 = B_{k1}(3,6) = ?$
$e_2 = B_{k1}(12,13) = ?$
$h_1 = B_{k1}(12,7) = ?$
$h_2 = B_{k1}(15,6) = ?$
$e_1 = B_{k1}(3,6) = 0, \quad S^* = \phi,$ dado que no se puede cortar ninguna pieza de dimensiones $(3,7),(8,4),(4,5)$ en el tablero con dimensiones $(3,6)$

Seguidamente calculamos e_2, de la siguiente manera:

$e_2 = B_{k1}(12,13).S^* = \{(3,7),(8,4),(4,5)\}$

Luego resolvemos el siguiente problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 78x_1 + 65x_2 + 28x_3 \\ \text{s.a} & 21x_1 + 32x_2 + 20x_3 \leq 156 \end{array}$$

$0 \leq x_1 \leq min\{6-0, [\frac{12}{3}][\frac{13}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6,4\} \Rightarrow 0 \leq x_1 \leq 4$
$0 \leq x_2 \leq min\{8-0, [\frac{12}{8}][\frac{13}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8,3\} \Rightarrow 0 \leq x_2 \leq 3$
$0 \leq x_3 \leq min\{10-0, [\frac{12}{4}][\frac{13}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10,6\} \Rightarrow 0 \leq x_3 \leq 6$

Hacemos:

$U_1 = min\{6,4\} = 4 \Rightarrow (4)(78) = 312 \Rightarrow z^* = 312, j^* = 1$
$U_2 = min\{8,3\} = 3 \Rightarrow (3)(65) = 130 \not> 312 \Rightarrow z^* = 312, j^* = 1$
$U_3 = min\{10,6\} = 6 \Rightarrow (6)(28) = 168 \not> 312 \Rightarrow z^* = 312, j^* = 1$

Realizamos cortes sucesivos

$x_1 \leq min\{4, [\frac{156}{21}]\} = 4$
$z = (4)(78) = 312$

$c = 156 - (4)(121) = 72$

$x_2 \leq min\{3, [\frac{72}{32}]\} = 2$
$z = 312 + (2)(65) = 442$
$c = 72 - (2)(32) = 8$

$x_3 \leq min\{6, [\frac{8}{20}]\} = 0$
$z = 442 + 0(20) = 442$
$c = 8$

Luego tenemos:

$z^* = 312 \not> 442 \Rightarrow z = 442, x_1 = 4, \quad x_2 = 2, \quad x_3 = 0$
$e_2 = B_{k1}(12, 13) = 442$

Inmediatamente pasamos a determinar el valor de h_1:

$h_1 = B_{k1}(12, 7), \quad S^* = \{(3, 7), (8, 4), (4, 5)\}$

Resolvemos el siguiente problema Knapsack

Maximizar	$78x_1 + 65x_2 + 28x_3$
s.a	$21x_1 + 32x_2 + 20x_3 \leq 84$

$0 \leq x_1 \leq min\{6 - 0, [\frac{12}{3}][\frac{7}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6, 4\} \Rightarrow 0 \leq x_1 \leq 4$
$0 \leq x_2 \leq min\{8 - 0, [\frac{12}{8}][\frac{7}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8, 1\} \Rightarrow 0 \leq x_2 \leq 1$
$0 \leq x_3 \leq min\{10 - 0, [\frac{12}{4}][\frac{7}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 3\} \Rightarrow 0 \leq x_3 \leq 3$

Hacemos:

$U_1 = min\{6, 4\} = 4 \Rightarrow (4)(78) = 312 \Rightarrow z^* = 312, j^* = 1$
$U_2 = min\{8, 1\} = 1 \Rightarrow (1)(65) = 65 \not> 312 \Rightarrow z^* = 312, j^* = 1$
$U_3 = min\{10, 3\} = 3 \Rightarrow (3)(28) = 84 \not> 312 \Rightarrow z^* = 312, j^* = 1$

Realizamos cortes sucesivos

$x_1 \leq min\{4, [\frac{84}{21}]\} = 4$
$z = (4)(78) = 312$
$c = 84 - (4)(21) = 0$

$x_2 \leq min\{1, [\frac{0}{32}]\} = 0$
$z = 312 + (0)(65) = 312$
$c = 0 - (0)(20)$

$x_3 \leq min\{3, [\frac{0}{20}]\} = 0$
$z = 312 + (0)(28) = 312$
$c = 0 - (0)(20)$

Luego tenemos:

$z^* = 312 \not> 312 \Rightarrow z = 312, x_1 = 4, \quad x_2 = 0, \quad x_3 = 0$
$h_1 = B_{k1}(12, 7) = 312$

De forma análoga, procedemos a calcular h_2

$h_2 = B_{k1}(15, 6), \quad s^* = \{(8, 4), (4, 5)\}$, dado que, de la pieza con dimensiones $(15, 6)$ no es posible cortar piezas con dimensiones $(3, 7)$

$$\begin{array}{ll} \text{Maximizar} & 65x_2 + 28x_3 \\ \text{s.a} & 32x_2 + 20x_3 \leq 90 \end{array}$$

$0 \leq x_1 \leq min\{8 - 0, [\frac{15}{8}][\frac{6}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{6, 1\} \Rightarrow 0 \leq x_2 \leq 1$
$0 \leq x_3 \leq min\{10 - 0, [\frac{15}{4}][\frac{6}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 3\} \Rightarrow 0 \leq x_3 \leq 3$

Hacemos:

$U_2 = min\{6, 1\} = 1 \Rightarrow (1)(65) \Rightarrow z^* = 65, j^* = 2$
$U_3 = min\{10, 3\} = 1 \Rightarrow (3)(28) = 84 > 65 \Rightarrow z^* = 84, j^* = 3$

Luego realizamos cortes sucesivos

$x_2 \leq min\{1, [\frac{90}{32}]\} = 1$
$z = (1)(65) = 65$
$c = 90 - (1)(32) = 58$

$x_3 \leq min\{3, [\frac{58}{20}]\} = 2$
$z = 65 + (2)(28) = 121$
$c = 58 - 40 = 18$

Luego tenemos:

$z^* = 84 \not> 121 \Rightarrow z = 121, x_1 = 0, \quad x_2 = 1, \quad x_3 = 2$
$h_2 = B_{k1}(12, 10) = 121$
$e_1 = B_{k1}(3, 6) = 0$
$e_2 = B_{k1}(12, 13) = 442$
$e_1 = B_{k1}(12, 7) = 312$
$h_2 = B_{k1}(15, 6) = 121$

Pieza	Corte	Estimador	Suma	$B_i = v_i + \max\{e_1 + e_2, h_1 + h_2\}$
(3,7)	Vertical	0	442	78+442=520
		442		
	Horizontal	312	433	
		121		

Tabla 5.5: Corte en dos dimensiones

Ahora, aplicamos el procedimiento antes expuesto para la Pieza (8,4), y de esta manera tenemos:

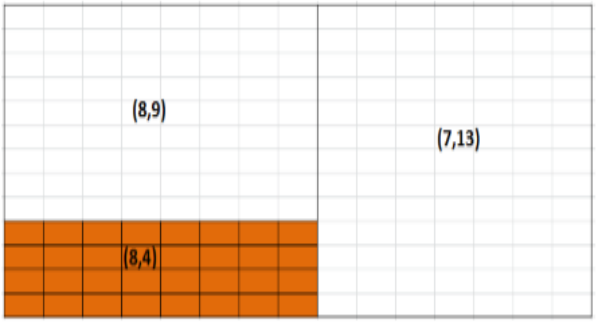


Figura 5.16: Tablero y piezas por cortar

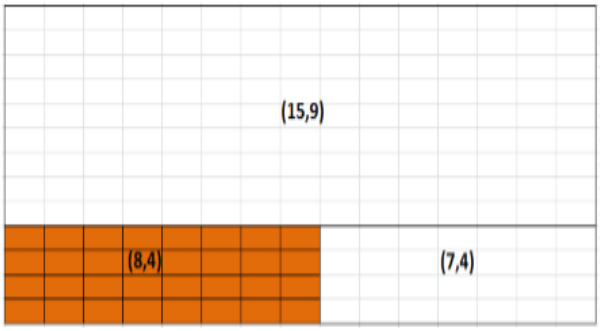


Figura 5.17: Tablero y piezas por cortar

Seguidamente realizamos el cálculo de los estimadores de los 4 tableros resultantes:

$e_1 = B_{k1}(8, 9) =?$
$e_2 = B_{k1}(7, 13) =?$
$h_1 = B_{k1}(7, 4) =?$
$h_2 = B_{k1}(15, 9) =?$

Para el estimador e_1 tenemos lo siguiente:

$e_1 = B_{k1}(8, 9)$
$S^* = (3, 7), (8, 4), (4, 5)$

Ahora resolvemos el problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 78x_1 + 65x_2 + 28x_3 \\ \text{s.a} & 21x_1 + 32x_2 + 20x_3 \leq 72 \end{array}$$

$0 \leq x_1 \leq min\{6 - 0, [\frac{8}{3}][\frac{9}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6, 2\} \Rightarrow 0 \leq x_1 \leq 2$
$0 \leq x_2 \leq min\{8 - 0, [\frac{8}{8}][\frac{9}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8, 2\} \Rightarrow 0 \leq x_2 \leq 2$
$0 \leq x_3 \leq min\{10 - 0, [\frac{8}{4}][\frac{9}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 2\} \Rightarrow 0 \leq x_3 \leq 2$

Posteriormente hacemos:

$U_1 = min\{6, 2\} = 2 \Rightarrow (2)(78) = 156 \Rightarrow z^* = 156, j^* = 1$
$U_2 = min\{8, 2\} = 2 \Rightarrow (2)(65) = 130 \not> 156 \Rightarrow z^* = 156, j^* = 1$
$U_3 = min\{10, 2\} = 2 \Rightarrow (2)(28) = 56 \not> 156 \Rightarrow z^* = 156, j^* = 1$

Luego de lo anterior realizamos los cortes sucesivos:

$x_1 \leq min\{2, [\frac{72}{21}]\} = 2$
$z = (2)(78) = 156$
$c = 72 - (2)(21) = 30$

$x_2 \leq min\{2, [\frac{30}{32}]\} = 0$
$z = 156 + (0)(65) = 156$
$c = 30 - (0)(32) = 30$

$x_3 \leq min\{2, [\frac{30}{20}]\} = 1$
$z = 156 + (1)(28) = 184$
$c = 30 - 20 = 10$

Finalmente tenemos que:

$z^* = 156 \not> 184 \Rightarrow z = 184, x_1 = 2, \quad x_2 = 0, \quad x_3 = 1$
$e_1 = B_{k1}(8, 6) = 65$

Ahora procederemos a calcular de forma similar el estimador e_2,

$e_2 = B_{k1}(7, 13), S^* = \{(3, 7), (4, 5)\}$

Planteamos y resolvemos el problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 78x_1 + 28x_3 \\ \text{s.a} & 21x_1 + 20x_3 \leq 91 \end{array}$$

$0 \leq x_1 \leq min\{6-0, [\frac{7}{3}][\frac{13}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6,2\} \Rightarrow 0 \leq x_1 \leq 2$
$0 \leq x_3 \leq min\{10-0, [\frac{7}{4}][\frac{13}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10,2\} \Rightarrow 0 \leq x_3 \leq 2$

Hacemos:

$U_1 = min\{4,2\} = 2 \Rightarrow (2)(78) = 156 \Rightarrow z^* = 156, j^* = 1$
$U_3 = min\{10,2\} = 2 \Rightarrow (2)(28) = 56 \not> 156 \Rightarrow z^* = 156, j^* = 1$

Seguidamente realizamos los cortes sucesivos

$x_1 \leq min\{2, [\frac{91}{21}]\} = 2$
$z = (2)(78) = 156$
$c = 91 - (2)(21) = 49$

$x_3 \leq min\{2, [\frac{49}{20}]\} = 2$
$z = 156 + (2)(28) = 212$
$c = 49 - 40 = 9$

Como consecuencia de lo anterior tenemos que:

$z^* = 156 \not> 212 \Rightarrow z = 212, x_1 = 2, \quad x_2 = 0, \quad x_3 = 2$
$e_2 = B_{k1}(7,13) = 212$

Luego de esto se realizarán los cálculos correspondientes para h_1 y h_2.

En el caso del primero tenemos que:

$h_1 = B_{k1}(7,4) = 0, \quad S^* = \phi,$ no se puede cortar ninguna pieza en $(3,6)$

En el caso del segundo, se tendría lo siguiente:

$h_2 = B_{k1}(15,9), \quad S^* = \{(3,7),(8,4),(4,5)\}$

Resolviendo nuevamente el problema Knapsack correspondiente,

Maximizar	$78x_1 + 65x_2 + 28x_3$
s.a	$21x_1 + 32x_2 + 20x_3 \leq 135$

$0 \leq x_1 \leq min\{6-0, [\frac{15}{3}][\frac{9}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6,5\} \Rightarrow 0 \leq x_1 \leq 5$
$0 \leq x_2 \leq min\{8-0, [\frac{15}{8}][\frac{9}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{6,2\} \Rightarrow 0 \leq x_2 \leq 2$
$0 \leq x_3 \leq min\{10-0, [\frac{15}{4}][\frac{9}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10,3\} \Rightarrow 0 \leq x_3 \leq 3$

Luego hacemos:

$U_1 = min\{6,5\} = 2 \Rightarrow (5)(78) = 390 \Rightarrow z^* = 390, j^* = 1$
$U_2 = min\{8,2\} = 2 \Rightarrow (2)(65) = 130 \not> 390 \Rightarrow z^* = 390, j^* = 1$

$U_3 = min\{10, 3\} = 3 \Rightarrow (3)(28) = 84 \not> 390 \Rightarrow z^* = 390, j^* = 1$

Seguidamente realizamos los cortes sucesivos

$x_1 \leq min\{5, [\frac{135}{21}]\} = 5$
$z = (5)(78) = 390$
$c = 135 - (5)(21) = 30$

$x_2 \leq min\{2, [\frac{30}{32}]\} = 0$
$z = 390 + (0)(65) = 390$
$c = 30 - 0 = 30$

$x_3 \leq min\{3, [\frac{30}{20}]\} = 1$
$z = 390 + (1)(28) = 418$
$c = 30 - 20 = 10$

Luego:

$z^* = 390 \not> 418 \Rightarrow z = 418, x_1 = 5, \quad x_2 = 0, \quad x_3 = 1$
$h_2 = B_{k1}(12, 10) = 418$

Resumiendo todo los cálculos realizados anteriormente, tenemos que:

$e_1 = B_{k1}(3, 3) = 184$
$e_2 = B_{k1}(12, 10) = 212$
$h_1 = B_{k1}(12, 7) = 0$
$h_2 = B_{k1}(3, 3) = 418$

Pieza	**Corte**	**Estimador**	**Suma**	$\mathbf{B_i = v_i + \max\{e_1 + e_2, h_1 + h_2\}}$
(8,4)	Vertical	184	396	65+418=483
		212		
	Horizontal	0	418	
		418		

Tabla 5.6: Corte en dos dimensiones

De forma análoga procedemos a trabajar con la Pieza $(4, 5)$

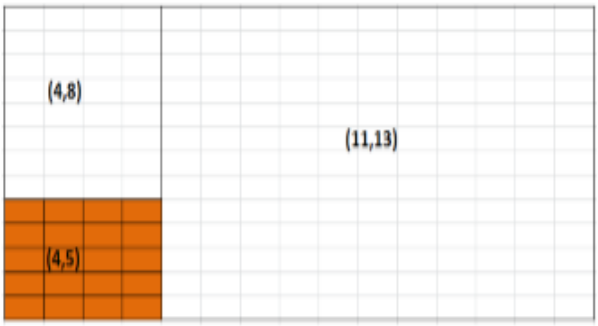


Figura 5.18: Tablero y piezas por cortar

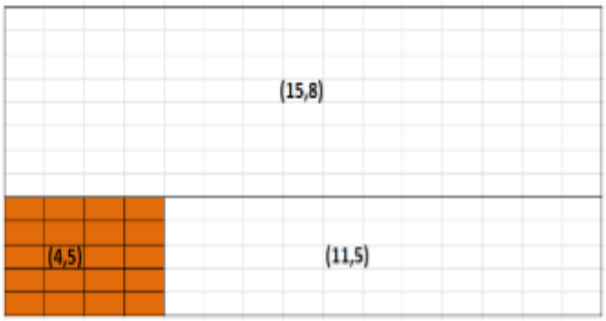


Figura 5.19: Tablero y piezas por cortar

Calculando los estimadores de los 4 tableros resultantes:

$e_1 = B_{k1}(4,8) = ?$
$e_2 = B_{k1}(11,13) = ?$
$h_1 = B_{k1}(15,8) = ?$
$h_2 = B_{k1}(11,5) = ?$
$e_1 = B_{k1}(4,8), \quad S^* = \{(3,7),(4,5)\}$

Resolvemos el problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 78x_1 + 28x_3 \\ \text{s.a} & 21x_1 + 20x_3 \leq 32 \end{array}$$

$0 \leq x_1 \leq min\{6-0, [\frac{4}{3}][\frac{8}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6,1\} \Rightarrow 0 \leq x_1 \leq 1$
$0 \leq x_3 \leq min\{10-0, [\frac{4}{4}][\frac{8}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10,1\} \Rightarrow 0 \leq x_3 \leq 1$

Hacemos:

$U_1 = min\{6,1\} = 1 \Rightarrow (1)(78) = 78 \Rightarrow z^* = 78, j^* = 1$

$U_3 = min\{10, 1\} = 1 \Rightarrow (1)(28) = 28 \not> 78 \Rightarrow z^* = 78, j^* = 1$

Después realizamos los cortes sucesivos

$x_1 \leq min\{1, [\frac{32}{21}]\} = 1$
$z = (1)(78) = 78$
$c = 32 - (1)(21) = 11$

$x_3 \leq min\{1, [\frac{11}{20}]\} = 0$
$z = 78 + (0)(28) = 78$
$c = 11 - 0 = 11$

Como producto de lo anterior tenemos que:

$z^* = 78 \not> 78 \Rightarrow z = 78, x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$
$e_1 = B_{k1}(4, 8) = 78$

Ahora procedemos el valor del siguiente estimador:

$e_2 = B_{k1}(11, 13), S^* = \{(3, 7), (8, 4), (4, 5)\}$

Formamos y resolvemos el problema Knapsack

Maximizar	$78x_1 + 65x_2 + 28x_3$
s.a	$21x_1 + 32x_2 + 20x_3 \leq 143$

$0 \leq x_1 \leq min\{6 - 0, [\frac{11}{3}][\frac{13}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6, 3\} \Rightarrow 0 \leq x_1 \leq 3$
$0 \leq x_2 \leq min\{8 - 0, [\frac{11}{8}][\frac{13}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8, 3\} \Rightarrow 0 \leq x_2 \leq 3$
$0 \leq x_3 \leq min\{10 - 0, [\frac{11}{4}][\frac{13}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 4\} \Rightarrow 0 \leq x_3 \leq 4$

Hacemos:

$U_1 = min\{6, 3\} = 3 \Rightarrow (3)(78) = 234 \Rightarrow z^* = 234, j^* = 1$
$U_2 = min\{8, 3\} = 3 \Rightarrow (3)(65) = 195 \not> 234 \Rightarrow z^* = 234, j^* = 1$
$U_3 = min\{10, 4\} = 4 \Rightarrow (4)(28) = 112 \not> 234 \Rightarrow z^* = 234, j^* = 1$

Luego realizamos los cortes sucesivos

$x_1 \leq min\{3, [\frac{143}{21}]\} = 3$
$z = (3)(78) = 234$
$c = 143 - (3)(21) = 80$

$x_2 \leq min\{3, [\frac{80}{32}]\} = 2$
$z = 234 + (2)(65) = 364$
$c = 80 - (2)(32) = 16$

$x_3 \leq min\{4, [\frac{16}{20}]\} = 0$
$z = 364 + (0)(28) = 364$
$c = 364 + 0 = 364$

Y de esta manera tenemos que:

$z^* = 234 \not> 364 \Rightarrow z = 364, x_1 = 3, \quad x_2 = 2, \quad x_3 = 0$
$e_2 = B_{k1}(11, 10) = 364$

De igual manera realizamos el cálculo para h_1:

$h_1 = B_{k1}(15, 8), \quad S^* = \{(3, 7), (8, 4), (4, 5)\}$

Nuevamente resolvemos el correspondiente problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 78x_1 + 65x_2 + 28x_3 \\ \text{s.a} & 21x_1 + 32x_2 + 20x_3 \leq 120 \end{array}$$

$0 \leq x_1 \leq min\{6 - 0, [\frac{15}{3}][\frac{8}{7}]\} \Rightarrow 0 \leq x_1 \leq min\{6, 5\} \Rightarrow 0 \leq x_1 \leq 5$
$0 \leq x_2 \leq min\{8 - 0, [\frac{15}{8}][\frac{8}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8, 2\} \Rightarrow 0 \leq x_2 \leq 2$
$0 \leq x_3 \leq min\{10 - 0, [\frac{15}{4}][\frac{8}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 3\} \Rightarrow 0 \leq x_3 \leq 3$

Hacemos:

$U_1 = min\{6, 5\} = 5 \Rightarrow (5)(78) = 390 \Rightarrow z^* = 390, j^* = 1$
$U_2 = min\{8, 2\} = 2 \Rightarrow (2)(65) = 130 \not> 390 \Rightarrow z^* = 390, j^* = 1$
$U_3 = min\{10, 3\} = 3 \Rightarrow (3)(28) = 84 \not> 390 \Rightarrow z^* = 390, j^* = 1$

En seguida realizamos los cortes sucesivos

$x_1 \leq min\{5, [\frac{120}{21}]\} = 5$
$z = (5)(78) = 390$
$c = 120 - (5)(21) = 15$

$x_2 \leq min\{2, [\frac{15}{32}]\} = 0$
$z = 390 + (0)(65) = 390$
$c = 15 - 0 = 15$

$x_3 \leq min\{3, [\frac{15}{20}]\} = 0$
$z = 390 + (0)(28) = 390$
$c = 390 + 0 = 390$

Y por tanto obtenemos lo siguiente:

$z^* = 390 \not> 390 \Rightarrow z = 390, x_1 = 5, \quad x_2 = 0, \quad x_3 = 0$
$h_1 = B_{k1}(15, 8) = 390$

Ahora pasamos a calcular de forma similar el valor de h_2

$h_2 = B_{k1}(11, 5), S^* = \{(8, 4), (4, 5)\}$

Resolvemos el problema Knapsack que corresponde,

$$\begin{array}{ll} \text{Maximizar} & 65x_1 + 28x_3 \\ \text{s.a} & 32x_1 + 20x_3 \leq 55 \end{array}$$

$0 \leq x_2 \leq min\{8 - 0, [\frac{11}{8}][\frac{5}{4}]\} \Rightarrow 0 \leq x_2 \leq min\{8, 1\} \Rightarrow 0 \leq x_2 \leq 2$
$0 \leq x_3 \leq min\{10 - 0, [\frac{11}{4}][\frac{5}{5}]\} \Rightarrow 0 \leq x_3 \leq min\{10, 2\} \Rightarrow 0 \leq x_3 \leq 2$

Hacemos:

$U_2 = min\{8, 1\} = 1 \Rightarrow (1)(65) = 65 \Rightarrow z^* = 65, j^* = 2$
$U_3 = min\{10, 2\} = 2 \Rightarrow (2)(28) = 56 \not> 65 \Rightarrow z^* = 65, j^* = 1$

Hacemos cortes sucesivos

$x_2 \leq min\{1, [\frac{55}{32}]\} = 1$
$z = (1)(65) = 65$
$c = 55 - 1(32) = 23$

$x_3 \leq min\{2, [\frac{23}{20}]\} = 1$
$z = 65 + (1)(28) = 93$
$c = 23 - 20 = 3$

Luego:

$z^* = 65 \not> 93 \Rightarrow z = 121, x_1 = 0, \quad x_2 = 1, \quad x_3 = 1$
$h_2 = B_{k1}(11, 5) = 93$

En resumen, como producto de todo el procedimiento anterior tenemos lo siguiente:

$e_1 = B_{k1}(4, 8) = 78$
$e_2 = B_{k1}(11, 13) = 299$
$h_1 = B_{k1}(15, 8) = 390$
$h_2 = B_{k1}(11, 5) = 93$

Pieza	Corte	Estimador	Suma	$B_i = v_i + \max\{e_1 + e_2, h_1 + h_2\}$
(4,5)	Vertical	78	442	28+483=511
		364		
	Horizontal	390	483	
		93		

Tabla 5.7: Corte en dos dimensiones

Etapa 2

$B = Max\{B_i, i \in I\}$
$B = Max\{520, 483, 511\} = 520$
$B > 0 \Rightarrow P = P \cup \{1\}, P = \{(3,7)\}, n_1 = 0 + 1 = 1, V_t = 78$
Si $(e_1 + e_2) \geq (h_1 + h_2), entonces R = R \cup \{(3,6)\} \cup \{12,13\}$
en caso contrario $R = R \cup \{(15,6)\} \cup \{12,7\}$

Port tanto la pieza cortada es la $(3,4)$ y los nuevos rectángulos a cortar son $(3,6)$ y $(12,6)$.

COTA SUPERIOR BK2
Cortamos la pieza $(3,7)$ tantas veces sea posible desde la parte inferior izquierda del tablero. Ahora vamos a analizar la mejor forma de cortar el tablero $(15,6)$

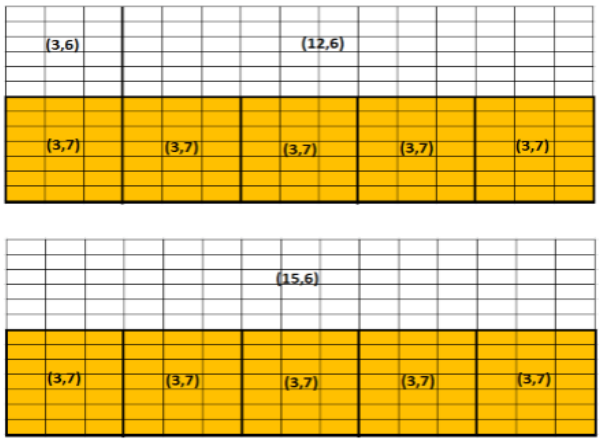


Figura 5.20: Tablero y piezas por cortar

PIEZA 1: Ya no se considera.
PIEZA 2: dimensiones (8,4)

$0 \leq x_i \leq min\{d_1 - n_1, [\frac{L_1}{l_i}][\frac{W_1}{w_i}] + [\frac{L_2}{l_i}][\frac{W_2}{w_i}], [\frac{L_3}{l_i}][\frac{W_3}{w_i}] + [\frac{L_4}{l_i}][\frac{W_4}{w_i}]\}$
$0 \leq x_2 \leq min\{6 - 0, max\{[\frac{3}{8}][\frac{6}{4}] + [\frac{12}{8}][\frac{6}{4}], [\frac{15}{8}][\frac{6}{4}] + [\frac{0}{8}][\frac{0}{4}]\}\}$
$0 \leq x_2 \leq min\{6, max\{0 + 1, 1 + 0\}\}$
$0 \leq x_2 \leq min\{6, max\{1, 1\}\} = min\{1, 1\} = 1$
$z = 390 + 1(65) = 455$

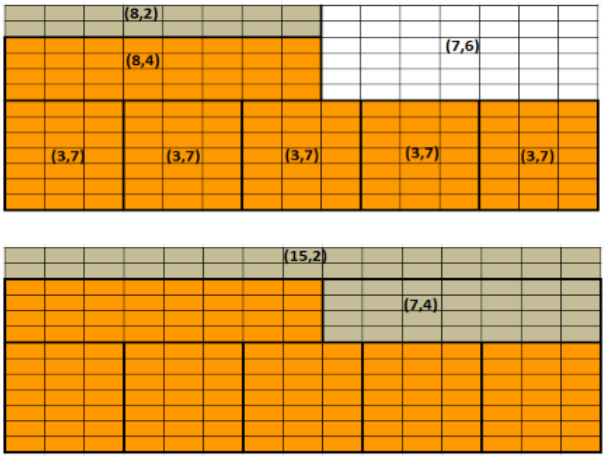


Figura 5.21: Tablero y piezas por cortar

PIEZA 3: dimensiones $(4, 5)$

$0 \leq x_3 \leq min\{8 - 0, max\{[\frac{8}{4}][\frac{2}{5}] + [\frac{7}{4}][\frac{6}{5}], [\frac{15}{4}][\frac{2}{5}] + [\frac{7}{4}][\frac{4}{5}]\}\}$
$0 \leq x_3 \leq min\{8, max\{0 + 1, 0 + 0\}\}$
$0 \leq x_3 \leq min\{8, max\{1, 0\}\} = min\{8, 1\} = 1$
$z = 455 + 1(65) = 483$

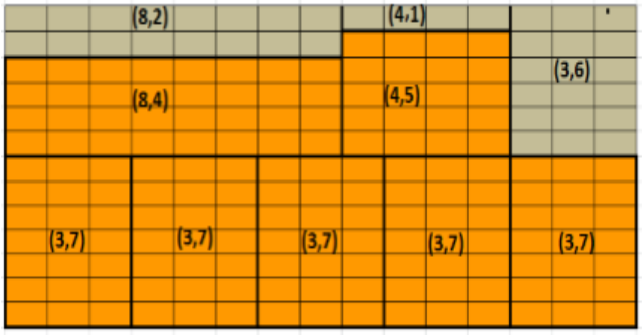


Figura 5.22: Tablero y piezas por cortar

5.4 Problemas Propuestos

1. Se hace un pedido a una papelería de 800 rollos de papel corrugado de 30 pulgadas de ancho, 500 rollos de 45 pulgadas de ancho y 1000 rollos de 56 pulgadas de ancho. Si la papelería tiene solamente rollos de 108 pulgadas de ancho. ¿cómo deben cortarse los rollos para surtir el pedido con el mínimo desperdicio de papel?

2. Una empresa produce bobinas de papel de 500 metros de longitud y un metro de ancho; se ha estimado que la demanda para el mes próximo es de 500 bobinas de 20 cm de ancho, 400 bobinas de 30 cm de ancho, 250 bobinas de 40 cm de ancho y 300 bobinas de 70 cm de ancho (todas las bobinas son de 500 metros de longitud).
   El fabricante debe cortar las bobinas de un metro de ancho con el tamaño de las peticiones para satisfacer la demanda, pero también desea que el desperdicio en el corte sea tal que el número de bobinas que fabrique de un metro sea mínimo con el objetivo que el costo de producción también lo sea, si considera desperdicio los sobrantes iguales o superiores a 10 cm.

3. Una empresa lamina y vende papel de aluminio de varios tamaños. Los clientes pueden ordenar rollos de papel de aluminio de 24 pulgadas, 20 pulgadas, 12 pulgadas u 8 pulgadas de ancho. La hoja estándar se fabrica en un ancho de 54 pulgadas y los tamaños más pequeños se cortan del rollo estándar.
   La empresa ha recibido los siguientes pedidos para el mes de julio: 330 rollos de 24 pulgadas de ancho, 120 rollos de 20 pulgadas de ancho, 480 rollos de 12 pulgadas de ancho y 160 rollos de 8 pulgadas de ancho. ¿Cómo debe cortar la empresa los rollos para cumplir con estos pedidos?

4. Una empresa de la industria papelera dispone de rollos de 100 cm, 80 cm y 55 cm y los clientes demandan 150 rollos de 45 cm, 200 rollos de 30 cm y 175 rollos de 18 cm. Lo que se quiere es minimizar la pérdida de material total que está dada por la ponderación de la pérdida en centímetros asociada a cada esquema de corte por la cantidad de veces que se utiliza el esquema respectivo.

		Posibles Combinaciones Tamaño de rollos (cm)			Pérdida
		45	30	18	cm
1	100cm	2	0	0	10
2		1	1	1	7
3		1	0	3	1
4		0	3	0	10
5		0	2	2	4
6		0	1	3	16
7		0	0	5	10
8	80cm	1	1	0	5
9		1	0	1	17
10		0	2	1	2
11		0	1	2	14
12		0	0	4	8
13	55cm	1	0	0	10
14		0	1	1	7
15		0	0	3	1
Demanda		150	200	175	

Tabla 5.8: *Datos del problema 4*

5. Dado un tablero cuyas dimensiones son 9 m de largo y 8 m de ancho. Se desea cortar en piezas más pequeñas con dimensiones 2m x 6m y 7m x 3m. Se desea determinar los tipos y cantidad de cortes que se deben hacer para optimizar los beneficios. A continuación se observa el tablero por cortar y los tipos de piezas demandados.

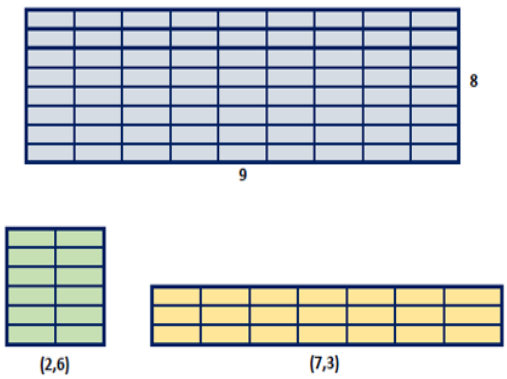


Figura 5.23: Tablero y piezas por cortar

Además, en la siguiente tabla se resume información referidas a las piezas demandadas:

$\mathbf{i}$	$(\mathbf{l_i}, \mathbf{w_i})$	$\mathbf{v_i}$	$\mathbf{s_i}$	$\mathbf{r_i} = \frac{\mathbf{v_i}}{\mathbf{s_i}}$	$\mathbf{d_i}$	$[\frac{\mathbf{L}}{\mathbf{l_i}}][\frac{\mathbf{W}}{\mathbf{w_i}}]$
1	$(2, 6)$	52	12	4.33	5	$[\frac{9}{2}][\frac{8}{6}] = 4$
2	$(7, 3)$	43	21	2.04	6	$[\frac{9}{7}][\frac{8}{3}] = 2$

Tabla 5.9: *Características de las piezas del problema*

5.5 Problemas resueltos

Problema 1
Solución:(Implementando Programación Lineal Entera con WinQsb)

Primeramente formamos todos los tipos de cortes posibles en papel de 108 pulgadas

Ancho	Tipos de corte en papel de 108					Requisito
	x_1	x_2	x_3	x_4	x_5	
30"	3	2	1	0	0	800
45"	0	1	0	2	1	500
56"	0	0	1	0	1	1000
Pérdida	18	3	22	18	7	

Tabla 5.10: *Información del problema 1*

El modelo matemático asociado al problema es:

$$\begin{aligned} \text{Minimizar} \quad & D = 3x_1 + 7x_2 + 22x_3 + 18x_4 + 18x_5 \\ \text{s.a} \quad & 3x_1 + 2x_3 + x_5 \geq 800 \\ & x_1 + 2x_2 + x_5 \geq 500 \\ & x_3 + x_5 \geq 1000 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

D: Desperdicio mínimo total
x_i: Cantidad de cortes realizados en los tipos

Una vez construido el modelo procederemos a resolverlo mediante Programación Lineal, auxiliándonos del WinQsb.

a. Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el Winqsb:

Figura 5.24: Módulo de Programación Lineal Entera del WinQsb

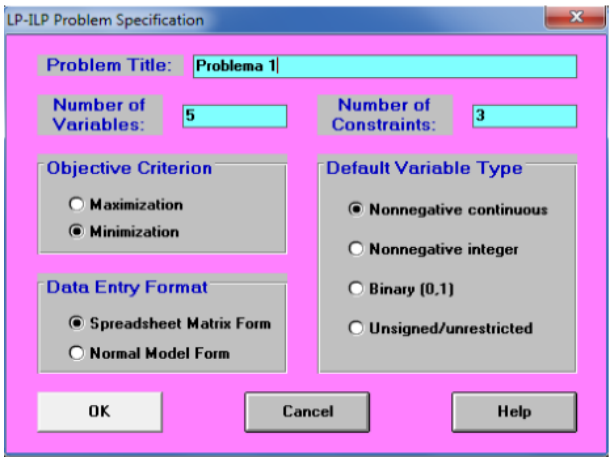


Figura 5.25: Datos del problema 1

b. Ingresamos el modelo relajado

Variable -->	X1	X2	X3	X4	X5	Direction	R. H. S.
Minimize	18	3	22	18	7		
C1	3	2	1	0	0	>=	800
C2	1	0	0	2	1	>=	500
C3	0	0	1	0	1	>=	1000
LowerBound	0	0	0	0	0		
UpperBound	M	M	M	M	M		
VariableType	ontinuou	ontinuou	ontinuou	ontinuou	ontinuou		

Figura 5.26: Datos del problema 1

c. Obtenemos la solución:

Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
X1	0	18.0000	0	13.5000	at bound	4.5000	M
X2	400.0000	3.0000	1,200.0000	0	basic	0	12.0000
X3	0	22.0000	0	13.5000	at bound	8.5000	M
X4	0	18.0000	0	18.0000	at bound	0	M
X5	1,000.0000	7.0000	7,000.0000	0	basic	0	20.5000
Objective	Function	(Min.) =	8,200.0000				

Figura 5.27: Solución del problema 1

La solución óptima del problema (0, 400, 0, 0, 1000) y con ella tenemos un valor óptimo de 8 200.

Problema 2
Solución:(Implementando Programación Lineal Entera con WinQsb)

Todos los tipos de cortes posibles se muestran en la siguiente tabla:

Patrones	**20**	**30**	**40**	**70**	**Sobrantes cm**
1	5	0	0	0	0
2	3	0	1	0	0
3	3	1	0	0	10
4	1	0	0	1	10
5	0	1	0	1	0
6	1	1	1	0	10
7	0	2	1	0	0
8	0	3	0	0	10
9	1	0	2	0	0
10	2	2	0	0	0

Tabla 5.11: *Datos del problema 2*

El modelo matemático asociado al problema es:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 10x_3 + 10x_4 + 10x_6 + 10x_8 \\
\text{s.a} \quad & 5x_1 + 3x_2 + 3x_3 + x_4 + x_6 + x_9 + 2x_{10} \geq 500 \\
& x_3 + x_5 + x_6 + 2x_7 + 3x_8 + 2x_{10} \geq 400 \\
& x_2 + x_6 + x_7 + 2x_9 \geq 250 \\
& x_4 + x_5 \geq 300 \\
& x_i \geq 0
\end{aligned}
$$

W: Función de costo del desperdicio en el corte de las bobinas
x_i: Número de bobinas a cortar de 500 m según el patrón i.
Una vez construido el modelo procederemos a resolverlo mediante Branch and Bound.

a. Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el Winqsb:

b. Ingresamos el modelo relajado

Variable -->	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Direction	R. H. S.
Minimize	0	0	10	10	0	10	0	10	0	0		
C1	5	3	3	1	0	1	0	0	1	2	>=	500
C2	0	0	1	0	1	1	2	3	0	2	>=	400
C3	0	1	0	0	0	1	1	0	2	0	>=	250
C4	0	0	0	1	1	0	0	0	0	0	>=	300
LowerBound	0	0	0	0	0	0	0	0	0	0		
UpperBound	M	M	M	M	M	M	M	M	M	M		
VariableType	ntinuc	ntinuc	ntinuc	ntinuo	ntinuc	ntinuc	ntinuc	ıtinu	ıtinu	ntinuo		

Figura 5.28: Datos del problema 2

c. Obtenemos la solución:

Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
X1	80.0000	0	0	0	basic	0	0
X2	0	0	0	0	at bound	0	M
X3	0	10.0000	0	10.0000	at bound	0	M
X4	0	10.0000	0	10.0000	at bound	0	M
X5	300.0000	0	0	0	basic	0	10.0000
X6	0	10.0000	0	10.0000	at bound	0	M
X7	50.0000	0	0	0	basic	0	0
X8	0	10.0000	0	10.0000	at bound	0	M
X9	100.0000	0	0	0	basic	0	0
X10	0	0	0	0	at bound	0	M
Objective	Function	(Min.) =	0	(Note:	Alternate	Solution	Exists!!)

Figura 5.29: Solución

La solución óptima del problema (80, 0, 0, 0, 300, 0, 50, 0, 100, 0) y con ella tenemos un valor óptimo de 0.

Problema 3

Solución:(Implementando Branch And Bound con WinQsb)

Método de corte													
Ancho(pulgadas)	1	2	3	4	5	6	7	8	9	10	11	12	13
24		1	1	1	1								
20	1					2	2	1	1	1			
12			1	2		1		2	1		4	2	
8		1	1		3		1	1	2	4		3	6
Desperdicio	6	2	2	6	6	2	2	2	6	2	6	6	6

Tabla 5.12: *Datos del problema 3*

El modelo matemático asociado al problema es:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 6x_1 + 2x_2 + 2x_3 + 6x_4 + 6x_5 + 2x_6 + 2x_7 + 2x_8 + 6x_9 + 2x_{10} \\
& +6x_{11} + 6x_{12} + 6x_{13} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_4 + x_5 \geq 320 \\
& x_1 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} \geq 120 \\
& x_3 + 2x_4 + x_6 + 2x_8 + x_9 + x_{11} + x_{12} \geq 480 \\
& x_2 + 2x_3 + 3x_5 + x_7 + x_8 + 2x_9 + 4x_{10} + 3x_{12} + 6x_{13} \geq 160 \\
& x_6 \geq 26 \\
& x_i \geq 0
\end{aligned}
$$

D: Desperdicio en el corte de papel

x_i: Cantidad de cortes hecho con el método i

a. Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el WinQsb:

Figura 5.30: Módulo de Programación Lineal del WinQsb

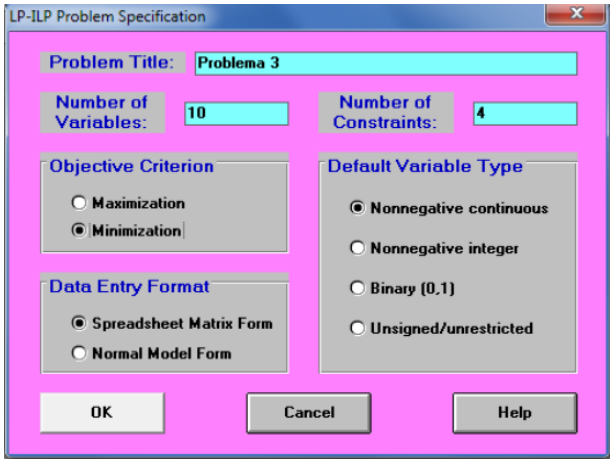


Figura 5.31: Información del problema 3

b. Ingresamos el modelo relajado

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	Direc	R. H. S.
Minimize	6	2	2	6	6	2	2	2	6	2	6	6	6		
C1	2	1	1	1	1	0	0	0	0	0	0	0	0	>=	320
C2	1	0	0	0	0	2	2	1	1	1	0	0	0	>=	120
C3	0	0	1	2	0	1	0	2	1	0	1	1	0	>=	480
C4	0	1	2	0	3	0	1	1	2	4	0	3	6	>=	160
owerBoun	0	0	0	0	0	0	0	0	0	0	0	0	0		
pperBoun	M	M	M	M	M	M	M	M	M	M	M	M	M		
ariableTyp	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu	tinu		

Figura 5.32: Datos del modelo del problema 3

c. Ingresamos el modelo relajado

Decision Variable	Solution Value	Unit Cost or	Total Contributior	Reduced Cost	Basis Status	Allowable	Allowable Max. c(j)
X1	0	6.0000	0	2.6667	at	3.3333	M
X2	0	2.0000	0	0.6667	at	1.3333	M
X3	320.0000	2.0000	640.0000	0	basic	0.6667	2.6667
X4	0	6.0000	0	3.3333	at	2.6667	M
X5	0	6.0000	0	4.6667	at	1.3333	M
X6	26.6667	2.0000	53.3333	0	basic	1.0000	2.5000
X7	0	2.0000	0	0.6667	at	1.3333	M
X8	66.6667	2.0000	133.3333	0	basic	1.0000	4.0000
X9	0	6.0000	0	4.6667	at	1.3333	M
X10	0	2.0000	0	1.3333	at	0.6667	M
X11	0	6.0000	0	5.3333	at	0.6667	M
X12	0	6.0000	0	5.3333	at	0.6667	M
X13	0	6.0000	0	6.0000	at	0	M
Objective	Function	(Min.) =	826.6666				

Figura 5.33: solución del modelo

Por tanto obtenemos nuestra primera solución al problema $(0, 0, 320, 0, 0, 26.66, 0, 66.66, 0, 0, 0, 0, 0)$ y $Z = 826.66$ Puesto que las soluciones no son enteras procedemos a ramificar tomando las variables cuyo valor no es entero, en este caso x_6.

Luego escogemos la variable x_6 y agregamos una rama en donde $x_6 \leq 26$, es decir la parte entera de 26.66

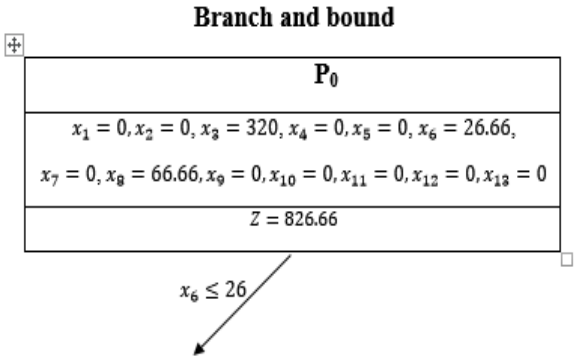


Figura 5.34: Solución P_0

Por lo tanto nuestro problema ahora queda de la siguiente forma:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 6x_1 + 2x_2 + 2x_3 + 6x_4 + 6x_5 + 2x_6 + 2x_7 + 2x_8 + 6x_9 + 2x_{10} \\
& \quad +6x_{11} + 6x_{12} + 6x_{13} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_4 + x_5 \geq 320 \\
& x_1 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} \geq 120 \\
& x_3 + 2x_4 + x_6 + 2x_8 + x_9 + x_{11} + x_{12} \geq 480 \\
& x_2 + 2x_3 + 3x_5 + x_7 + x_8 + 2x_9 + 4x_{10} + 3x_{12} + 6x_{13} \geq 160 \\
& x_6 \geq 26 \\
& x_i \geq 0
\end{aligned}
$$

Resolviendo este problema mediante el WinQsb, tenemos:

Decision Variable	Solution Value	Unit Cost or	Total Contributio	Reduced	Basis Status	Allowable	Allowable
X1	0	6.0000	0	2.0000	at bound	4.0000	M
X2	0	2.0000	0	0.5000	at bound	1.5000	M
X3	320.0000	2.0000	640.0000	0	basic	0.5000	2.5000
X4	0	6.0000	0	3.5000	at bound	2.5000	M
X5	0	6.0000	0	4.5000	at bound	1.5000	M
X6	26.0000	2.0000	52.0000	0	basic	-M	2.5000
X7	0.5000	2.0000	1.0000	0	basic	1.3333	4.0000
X8	67.0000	2.0000	134.0000	0	basic	1.0000	5.0000
X9	0	6.0000	0	4.5000	at bound	1.5000	M
X10	0	2.0000	0	1.0000	at bound	1.0000	M
X11	0	6.0000	0	5.5000	at bound	0.5000	M
X12	0	6.0000	0	5.5000	at bound	0.5000	M
X13	0	6.0000	0	6.0000	at bound	0	M
Objective	Function	(Min.) =	827.0000				

Figura 5.35: Solución del WinQsb

Es decir que la solución para dicho modelo es $(0, 0, 320, 0, 0, 26.66, 0.5, 67, 0, 0, 0, 0, 0)$ y $Z = 827$. Luego procedemos a generar la rama que corresponde:

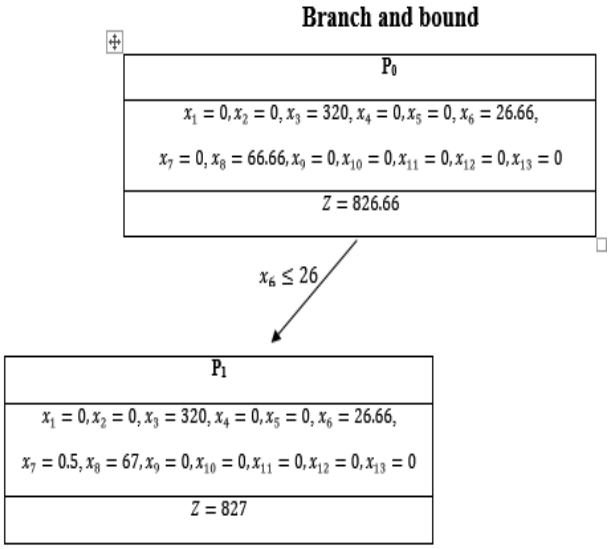


Figura 5.36: Solución P_0, P_1

Como se puede apreciar, tenemos una solución con variables no enteras y que además generan un valor de la función objetivo superior a la solución del problema inicial. Posteriormente, luego procedemos a la rama al lado derecho y agregamos la restricción $x_6 \geq 27$. Y en este caso obtenemos el problema inicial más esta última restricción:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 6x_1 + 2x_2 + 2x_3 + 6x_4 + 6x_5 + 2x_6 + 2x_7 + 2x_8 + 6x_9 + 2x_{10} \\
& \quad +6x_{11} + 6x_{12} + 6x_{13} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_4 + x_5 \geq 320 \\
& x_1 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} \geq 120 \\
& x_3 + 2x_4 + x_6 + 2x_8 + x_9 + x_{11} + x_{12} \geq 480 \\
& x_2 + 2x_3 + 3x_5 + x_7 + x_8 + 2x_9 + 4x_{10} + 3x_{12} + 6x_{13} \geq 160 \\
& x_6 \geq 27 \\
& x_i \geq 0
\end{aligned}
$$

Seguidamente resolvemos el modelo y tenemos la siguiente tabla con la solución

Decision Variable	Solution Value	Unit Cost or	Total Contributio	Reduced	Basis Status	Allowable	Allowable
X1	0	6.0000	0	4.0000	at bound	2.0000	M
X2	0	2.0000	0	1.0000	at bound	1.0000	M
X3	320.0000	2.0000	640.0000	0	basic	1.0000	3.0000
X4	0	6.0000	0	3.0000	at bound	3.0000	M
X5	0	6.0000	0	5.0000	at bound	1.0000	M
X6	27.0000	2.0000	54.0000	0	basic	1.0000	M
X7	0	2.0000	0	2.0000	at bound	0.0000	M
X8	66.5000	2.0000	133.0000	0	basic	0	4.0000
X9	0	6.0000	0	5.0000	at bound	1.0000	M
X10	0	2.0000	0	2.0000	at bound	0	M
X11	0	6.0000	0	5.0000	at bound	1.0000	M
X12	0	6.0000	0	5.0000	at bound	1.0000	M
X13	0	6.0000	0	6.0000	at bound	0	M
Objective	Function	(Min.) =	827.0000				

Figura 5.37: Solución del WinQsb

En este caso obtenemos como solución $(0, 0, 320, 0, 0, 27, 0, 66.5, 0, 0, 0, 0, 0)$ y $Z = 827$. Creando la rama que corresponde tendríamos la gráfica:

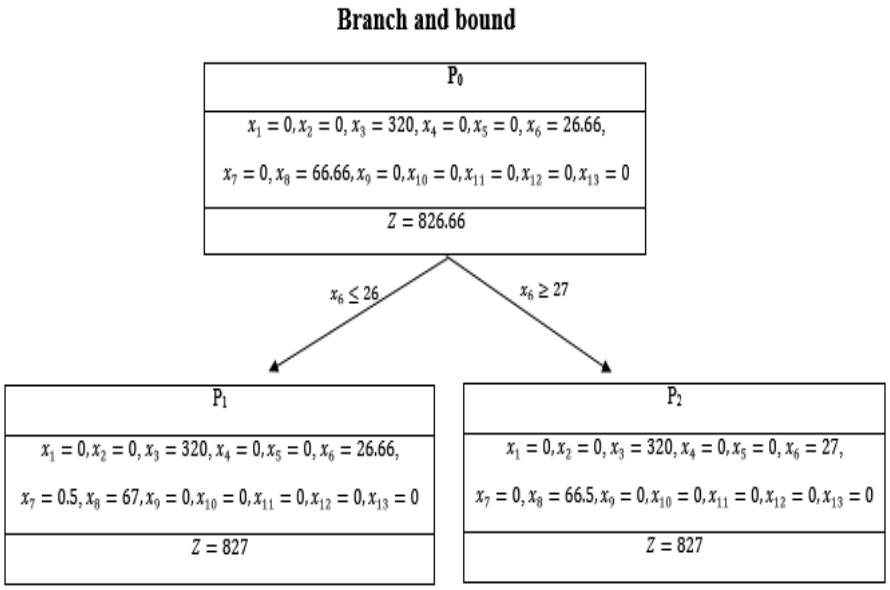


Figura 5.38: Solución $P_0, P_1, , P_2$

Como se puede apreciar, en ambas ramas tenemos una solución con variables no enteras y que además generan un valor de la función objetivo superior a la solución del problema inicial. Luego elegimos la rama al lado derecho y agregamos la restricción $x_8 \leq 66$. Y en este caso obtenemos al modelo le agregamos esta última restricción:

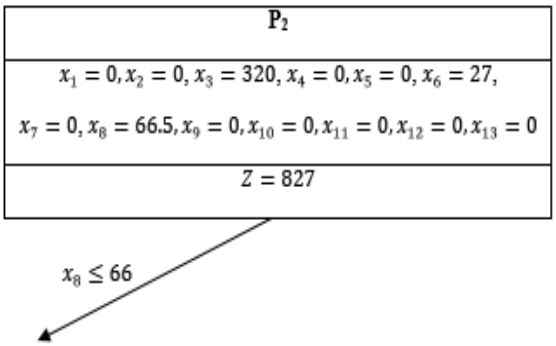


Figura 5.39: Solución P_2

El modelo sería el siguiente:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 6x_1 + 2x_2 + 2x_3 + 6x_4 + 6x_5 + 2x_6 + 2x_7 + 2x_8 + 6x_9 + 2x_{10} \\
& \quad +6x_{11} + 6x_{12} + 6x_{13} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_4 + x_5 \geq 320 \\
& x_1 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} \geq 120 \\
& x_3 + 2x_4 + x_6 + 2x_8 + x_9 + x_{11} + x_{12} \geq 480 \\
& x_2 + 2x_3 + 3x_5 + x_7 + x_8 + 2x_9 + 4x_{10} + 3x_{12} + 6x_{13} \geq 160 \\
& x_6 \geq 27 \\
& x_8 \geq 66 \\
& x_i \geq 0
\end{aligned}
$$

Resolviendo tenemos:

Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
X1	0	6.0000	0	6.0000	at bound	0	M
X2	0	2.0000	0	2.0000	at bound	0	M
X3	321.0000	2.0000	642.0000	0	basic	1.0000	2.0000
X4	0	6.0000	0	2.0000	at bound	4.0000	M
X5	0	6.0000	0	6.0000	at bound	0	M
X6	27.0000	2.0000	54.0000	0	basic	2.0000	M
X7	0	2.0000	0	2.0000	at bound	0.0000	M
X8	66.0000	2.0000	132.0000	0	basic	-M	4.0000
X9	0	6.0000	0	4.0000	at bound	2.0000	M
X10	0	2.0000	0	2.0000	at bound	0	M
X11	0	6.0000	0	4.0000	at bound	2.0000	M
X12	0	6.0000	0	4.0000	at bound	2.0000	M
X13	0	6.0000	0	6.0000	at bound	0	M
Objective	Function	(Min.) =	828.0000	(Note:	Alternate	Solution	Exists!!)

Figura 5.40: Solución del WinQsb

Obtenemos la siguiente solución $(0, 0, 321, 0, 0, 27, 0, 66, 0, 0, 0, 0, 0)$ y $Z = 828$. Creando la rama que corresponde tendríamos la gráfica:

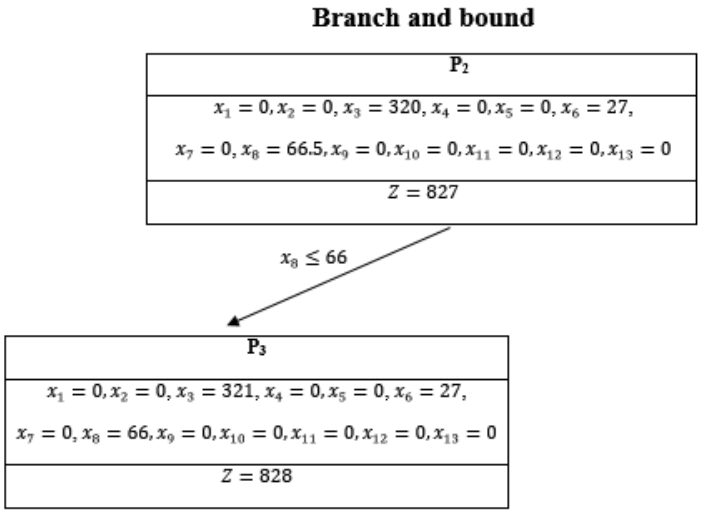


Figura 5.41: Solución P_2, P_3

Luego procedemos ramificar al lado derecho y agregamos la restricción $x_8 \geq 67$. Y en este caso el modelo toma la forma:

$$
\begin{aligned}
\text{Minimizar} \quad & W = 6x_1 + 2x_2 + 2x_3 + 6x_4 + 6x_5 + 2x_6 + 2x_7 + 2x_8 + 6x_9 + 2x_{10} \\
& \quad +6x_{11} + 6x_{12} + 6x_{13} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_4 + x_5 \geq 320 \\
& x_1 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} \geq 120 \\
& x_3 + 2x_4 + x_6 + 2x_8 + x_9 + x_{11} + x_{12} \geq 480 \\
& x_2 + 2x_3 + 3x_5 + x_7 + x_8 + 2x_9 + 4x_{10} + 3x_{12} + 6x_{13} \geq 160 \\
& x_6 \geq 27 \\
& x_8 \geq 67 \\
& x_i \geq 0
\end{aligned}
$$

De aquí obtenemos la siguiente solución:

Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
X1	0	6.0000	0	2.0000	at bound	4.0000	M
X2	1.0000	2.0000	2.0000	0	basic	1.0000	2.0000
X3	319.0000	2.0000	638.0000	0	basic	2.0000	3.0000
X4	0	6.0000	0	4.0000	at bound	2.0000	M
X5	0	6.0000	0	4.0000	at bound	2.0000	M
X6	27.0000	2.0000	54.0000	0	basic	0	M
X7	0	2.0000	0	2.0000	at bound	0	M
X8	67.0000	2.0000	134.0000	0	basic	0	M
X9	0	6.0000	0	6.0000	at bound	0	M
X10	0	2.0000	0	2.0000	at bound	0	M
X11	0	6.0000	0	6.0000	at bound	0	M
X12	0	6.0000	0	6.0000	at bound	0	M
X13	0	6.0000	0	6.0000	at bound	0	M
Objective	Function	(Min.) =	828.0000	(Note:	Alternate	Solution	Exists!!)

Figura 5.42: Solución del WinQsb

Obtenemos la siguiente solución $(0, 1, 319, 0, 0, 27, 0, 67, 0, 0, 0, 0, 0)$ y $Z = 828$. Creando la rama que corresponde tendríamos la gráfica:

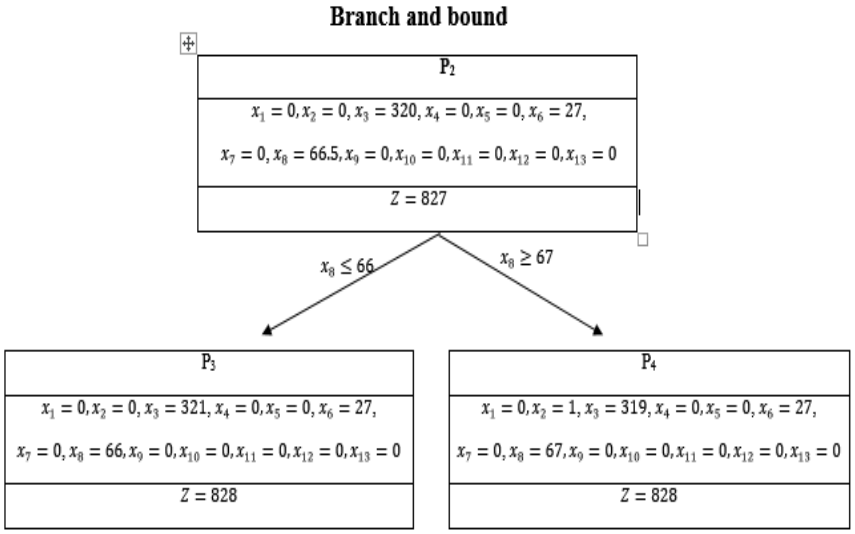


Figura 5.43: Solución $P_2, P_3, , P_4$

Por tanto la solución es $(0, 1, 319, 0, 0, 27, 0, 67, 0, 0, 0, 0, 0)$ y $Z = 828$.

Problema 4
Solución:(Implementando Programación Lineal Entera con WinQsb)

A continuación se muestran los posibles cortes:

		Posibles Combinaciones Tamaño de rollos (cm)			Pérdida
		45	30	18	cm
1	100cm	2	0	0	10
2		1	1	1	7
3		1	0	3	1
4		0	3	0	10
5		0	2	2	4
6		0	1	3	16
7		0	0	5	10
8	80cm	1	1	0	5
9		1	0	1	17
10		0	2	1	2
11		0	1	2	14
12		0	0	4	8
13	55cm	1	0	0	10
14		0	1	1	7
15		0	0	3	1
Demanda		150	200	175	

Tabla 5.13: *Datos del problema 4*

El modelo asociado al problema será:

$$\begin{aligned}
\text{Minimizar} \quad & D = 10x_1 + 7x_2 + x_3 + 10x_4 + 4x_5 + 16x_6 + 10x_7 + 5x_8 + 17x_9 + 2x_{10} \\
& \quad +14x_{11} + 6x_{12} + 10x_{13} + 7x_{14} + x_{15} \\
\text{s.a} \quad & 2x_1 + x_2 + x_3 + x_8 + x_9 + x_{13} \geq 150 \\
& x_2 + 3x_4 + 2x_5 + x_6 + x_8 + 2x_{10} + x_{11} + x_{14} \geq 200 \\
& x_2 + 3x_3 + 2x_5 + 3x_6 + 5x_7 + x_9 + x_{10} + 2x_{11} + 4x_{12} + x_{14} + 3_{15} \geq 175 \\
& x_i \geq 0
\end{aligned}$$

D: Función de costo del desperdicio en el corte de los rollos
x_i: Número de rollos a cortar bajo el patrón i.

Una vez construido el modelo procederemos a resolverlo mediante Programación Lineal, particularmente aplicando el Branch and Bound.

a. Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el Winqsb:

Figura 5.44: Módulo de Programación Lineal Entera del WinQsb

b. Ingresamos el modelo relajado

Variable	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	)irectio	R. H.
Minimize	10	7	1	10	4	16	10	5	17	2	14	8	10	7	1		
C1	2	1	1	0	0	0	0	1	1	0	0	0	1	0	0	>=	150
C2	0	1	3	0	2	1		1	0	2	1	0	0	1	0	>=	200
C3	0	1	3	0	2	3	5	0	1	1	2	4	0	1	3	>=	175
owerBoun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
pperBoun	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M		
ariableTyp	ntinuo	ntinuc	ntinuc	ntinuc	ntinuo	ntinuo	ntinuc	ntinuc	ntinuc	ntinuc	ntinuc	ntinuc	ntinu	ntinu	ntinuc		

Figura 5.45: Datos del modelo

c. Obtenemos la solución:

Decision Variable	Solution Value	Unit Cost or	Total ontributio	Reduced Cost	Basis Status	Allowable	Allowable Max. c[j]
X1	0	10.0000	0	8.0000	at bound	2.0000	M
X2	0	7.0000	0	6.0000	at bound	1.0000	M
X3	150.0000	1.0000	150.0000	0	basic	0	5.0000
X4	0	10.0000	0	10.0000	at bound	0	M
X5	0	4.0000	0	4.0000	at bound	0	M
X6	0	16.0000	0	16.0000	at bound	0.0000	M
X7	0	10.0000	0	10.0000	at bound	0.0000	M
X8	0	5.0000	0	4.0000	at bound	1.0000	M
X9	0	17.0000	0	16.0000	at bound	1.0000	M
X10	0	2.0000	0	2.0000	at bound	0	M
X11	0	14.0000	0	14.0000	at bound	0	M
X12	0	8.0000	0	8.0000	at bound	0.0000	M
X13	0	10.0000	0	9.0000	at bound	1.0000	M
X14	0	7.0000	0	7.0000	at bound	0	M
X15	0	1.0000	0	1.0000	at bound	0.0000	M
Objective	Function	(Min.) =	150.0000				

Figura 5.46: Solución del modelo

La solución óptima del problema $(0, 0, 150, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ y con ella tenemos un valor óptimo de 150.

Ejercicio 5
Solución: Implementado un heurístico para el corte de piezas

Dado un tablero cuyas dimensiones son 9 m de largo y 8 m de ancho. Se desea cortar en piezas más pequeñas con dimensiones 2m x 6m y 7m x 3m. Se desea determinar los tipos y cantidad de cortes que se deben hacer para optimizar los beneficios. A continuación se observa el tablero por cortar y los tipos de piezas demandados.

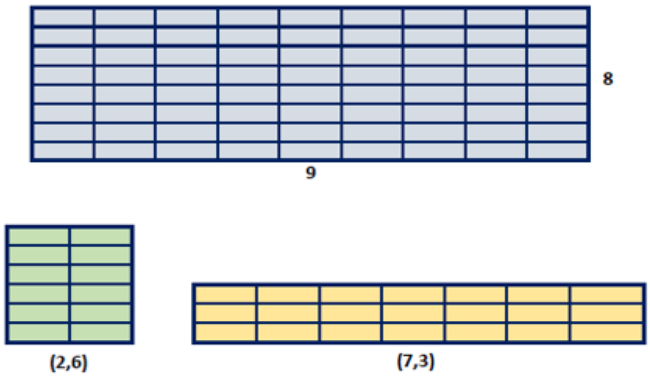


Figura 5.47: Tablero y piezas por cortar

Además, en la siguiente tabla se resume información referidas a las piezas demandadas:

$\mathbf{i}$	$(\mathbf{l_i}, \mathbf{w_i})$	$\mathbf{v_i}$	$\mathbf{s_i}$	$\mathbf{r_i} = \frac{\mathbf{v_i}}{\mathbf{s_i}}$	$\mathbf{d_i}$	$[\frac{\mathbf{L}}{\mathbf{l_i}}][\frac{\mathbf{W}}{\mathbf{w_i}}]$
1	$(2, 6)$	52	12	4.33	5	$[\frac{9}{2}][\frac{8}{6}] = 4$
2	$(7, 3)$	43	21	2.04	6	$[\frac{9}{7}][\frac{8}{3}] = 2$

Tabla 5.14: *Características de las piezas del problema 5*

A continuación se expone la implementación de cada una de las etapas del algoritmo:

Etapa 0

$R = \{(9, 8)\}$ Rectángulo original

$P = \{\} = \phi$ Conjunto de piezas cortadas

$V_T = 0$ Valor total de piezas cortadas

$S = \{(l_i, w_i) : i \in I\} = \{(2, 6), (7, 3)\}$ Piezas que se deben cortar

$n_1 = n_2 = n_3 = 0$ Número de piezas cortadas de cada tipo

Etapa 1

$R_k = \{(9, 8)\}$

$\{(2, 6), (7, 3)\}$ Piezas que se pueden cortar en R_k

Iniciamos el procedimiento para la Pieza $(2, 6)$

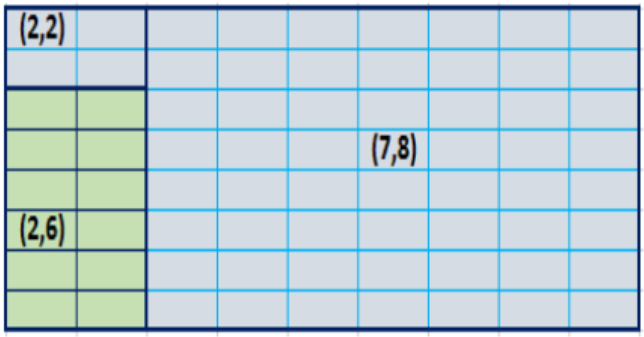


Figura 5.48: Tablero y piezas por cortar

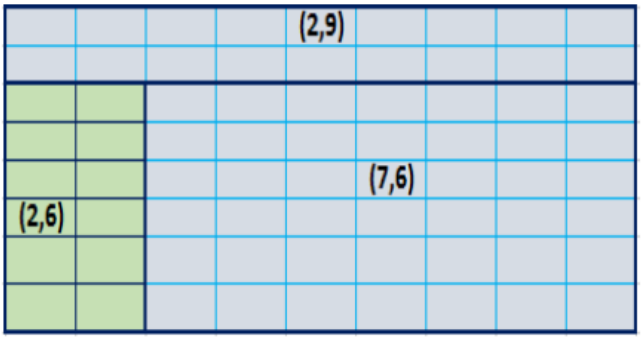


Figura 5.49: Tablero y piezas por cortar

Ahora procedemos a calcular los estimadores de los 4 tableros resultantes, es decir:

$e_1 = B_{k1}(2,2) = ?$
$e_2 = B_{k1}(7,8) = ?$
$h_1 = B_{k1}(7,6) = ?$
$h_2 = B_{k1}(2,9) = ?$
$e_1 = B_{k1}(2,2) = 0, \quad S^* = \phi,$ dado que no se puede cortar ninguna pieza con dimensiones $(2,6),(7,3)$ del tablero con dimensiones $(2,2)$

Seguidamente calculamos e_2, de la siguiente manera:

$e_2 = B_{k1}(7,8).S^* = \{(2,6),(7,3)\}$

Luego resolvemos el siguiente problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 52x_1 + 43x_2 \\ \text{s.a} & 12x_1 + 21x_2 \leq 156 \end{array}$$

$0 \leq x_1 \leq min\{5-0, [\frac{7}{2}][\frac{8}{6}]\} \Rightarrow 0 \leq x_1 \leq min\{5,3\} \Rightarrow 0 \leq x_1 \leq 3$
$0 \leq x_2 \leq min\{6-0, [\frac{7}{7}][\frac{8}{3}]\} \Rightarrow 0 \leq x_2 \leq min\{6,2\} \Rightarrow 0 \leq x_2 \leq 2$

Hacemos:

$U_1 = min\{5,3\} = 3 \Rightarrow (3)(52) = 156 \Rightarrow z^* = 156, j^* = 1$
$U_2 = min\{6,2\} = 2 \Rightarrow (2)(43) = 86 \not> 156 \Rightarrow z^* = 156, j^* = 1$

Realizamos cortes sucesivos

$x_1 \leq min\{3, [\frac{56}{12}]\} = 3$
$z = (3)(52) = 156$
$c = 56 - (3)(12) = 20$

$x_2 \leq min\{2, [\frac{20}{21}]\} = 0$

$z = 156 + (0)(21) = 156$
$c = 20 - (0)(21)(32) = 20$

Luego tenemos:

$z^* = 156 \not> 156 \Rightarrow z = 156, x_1 = 3, \quad x_2 = 0$
$e_2 = B_{k1}(7, 8) = 156$

Inmediatamente pasamos a determinar el valor de h_1:

$h_1 = B_{k1}(7, 6), \quad S^* = \{(2, 6), (7, 3)\}$

Resolvemos el siguiente problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 52x_1 + 43x_2 \\ \text{s.a} & 12x_1 + 21x_2 \leq 42 \end{array}$$

$0 \leq x_1 \leq min\{5 - 0, [\frac{7}{2}][\frac{6}{6}]\} \Rightarrow 0 \leq x_1 \leq min\{5, 3\} \Rightarrow 0 \leq x_1 \leq 3$
$0 \leq x_2 \leq min\{6 - 0, [\frac{7}{7}][\frac{6}{3}]\} \Rightarrow 0 \leq x_2 \leq min\{6, 2\} \Rightarrow 0 \leq x_2 \leq 2$

Hacemos:

$U_1 = min\{5, 3\} = 3 \Rightarrow (3)(52) = 156 \Rightarrow z^* = 156, j^* = 1$
$U_2 = min\{6, 2\} = 2 \Rightarrow (2)(43) = 86 \not> 156 \Rightarrow z^* = 156, j^* = 1$

Realizamos cortes sucesivos de la siguiente manera

$x_1 \leq min\{3, [\frac{42}{12}]\} = 3$
$z = (3)(52) = 156$
$c = 42 - (3)(12) = 6$

$x_2 \leq min\{2, [\frac{6}{21}]\} = 0$
$z = 156 + (0)(43) = 156$
$c = 6 - (0)(21) = 6$

Luego tenemos:

$z^* = 156 \not> 156 \Rightarrow z = 156, x_1 = 3, \quad x_2 = 0$
$h_1 = B_{k1}(7, 6) = 156$

De forma análoga, procedemos a calcular h_2

$h_2 = B_{k1}(2, 9), \quad s^* = \{(2, 6)\}$

Nuevamente pasamos a resolver el problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 52x_2 \\ \text{s.a} & 32x_1 \leq 18 \end{array}$$

$0 \leq x_1 \leq min\{5-0, [\frac{2}{2}][\frac{9}{6}]\} \Rightarrow 0 \leq x_1 \leq min\{5,1\} \Rightarrow 0 \leq x_2 \leq 1$

Hacemos:

$U_1 = min\{5,1\} = 1 \Rightarrow (1)(52) \Rightarrow z^* = 52, j^* = 1$

Luego realizamos cortes sucesivos

$x_1 \leq min\{1, [\frac{18}{12}]\} = 1$
$z = (1)(52) = 52$
$c = 18 - (1)(12) = 6$

$x_3 \leq min\{3, [\frac{58}{20}]\} = 2$
$z = 65 + (2)(28) = 121$
$c = 58 - 40 = 18$

Luego tenemos:

$z^* = 52 \not> 52 \Rightarrow z = 52, x_1 = 0, \quad x_2 = 0$
$h_2 = B_{k1}(2,9) = 52$

Como producto de todo el procedimiento expuesto anteriormente tenemos que

$e_1 = B_{k1}(2,2) = 0$
$e_2 = B_{k1}(7,8) = 156$
$e_1 = B_{k1}(7,6) = 156$
$h_2 = B_{k1}(2,9) = 52$

Pieza	**Corte**	**Estimador**	**Suma**	$\mathbf{B_i = v_i + \max\{e_1 + e_2, h_1 + h_2\}}$
(2,6)	Vertical	0	156	52+208=260
		156		
	Horizontal	156	208	
		52		

Tabla 5.15: Corte en dos dimensiones

Ahora, aplicamos el procedimiento antes expuesto para la Pieza $(7,3)$, y de esta manera tenemos:

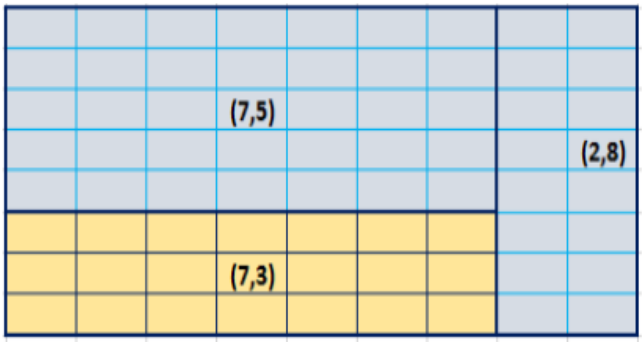


Figura 5.50: Tablero y piezas por cortar

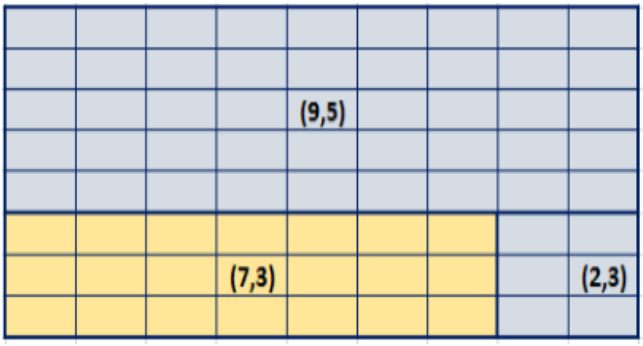


Figura 5.51: Tablero y piezas por cortar

Seguidamente realizamos el cálculo de los estimadores de los 4 tableros resultantes:

$e_1 = B_{k1}(7,5) =?$
$e_2 = B_{k1}(2,8) =?$
$h_1 = B_{k1}(2,3) =?$
$h_2 = B_{k1}(9,5) =?$

Para el estimador e_1 tenemos lo siguiente:

$e_1 = B_{k1}(7,5)$
$S^* = (7,3)$

Ahora resolvemos el problema Knapsack

$$\begin{array}{ll} \text{Maximizar} & 43x_2 \\ \text{s.a} & 21x_2 \leq 35 \end{array}$$

$0 \leq x_2 \leq min\{6-0, [\frac{7}{7}][\frac{5}{3}]\} \Rightarrow 0 \leq x_2 \leq 1$

Posteriormente hacemos:

$U_1 = min\{6, 1\} = 1 \Rightarrow (1)(43) = 43 \Rightarrow z^* = 43, j^* = 1$

Luego de lo anterior realizamos los cortes sucesivos:

$x_2 \leq min\{2, [\frac{35}{21}]\} = 1$
$z = (1)(43) = 43$
$c = 35 - (1)(21) = 14$

Finalmente tenemos que:

$z^* = 43 \not> 43 \Rightarrow z = 43, x_1 = 0, \quad x_2 = 1$
$e_1 = B_{k1}(7, 5) = 43$

Ahora procederemos a calcular de forma similar el estimador e_2,

$e_2 = B_{k1}(2, 8), S^* = \{(2, 6)\}$

Planteamos y resolvemos el problema Knapsack

Maximizar	$52x_1$
s.a	$12x_1 \leq 16$

$0 \leq x_1 \leq min\{5 - 0, [\frac{2}{2}][\frac{8}{6}]\} \Rightarrow 0 \leq x_1 \leq 1$

Hacemos:

$U_1 = min\{5, 1\} = 1 \Rightarrow (1)(52) = 52 \Rightarrow z^* = 52, j^* = 1$

Seguidamente realizamos los cortes sucesivos

$x_1 \leq min\{1, [\frac{16}{12}]\} = 1$
$z = (1)(52) = 52$
$c = 16 - (1)(12) = 4$

Como consecuencia de lo anterior tenemos que:

$z^* = 52 \not> 52 \Rightarrow z = 52, x_1 = 1, \quad x_2 = 0$
$e_2 = B_{k1}(2, 8) = 52$

Luego de esto se realizarán los cálculos correspondientes para h_1 y h_2.

En el caso del primero tenemos que:

$h_1 = B_{k1}(2, 3) = 0, \quad S^* = \phi,$ no se puede cortar ninguna pieza en $\{(2, 6), (7, 3)\}$

En el caso del segundo, se tendría lo siguiente:

$h_2 = B_{k1}(9,5), \quad S^* = \phi$ no se puede cortar ninguna pieza en $\{(2,6),(7,3)\}$

Pieza	Corte	Estimador	Suma	$B_i = v_i + \max\{e_1 + e_2, h_1 + h_2\}$
(7,3)	Vertical	43	95	43+95=138
		52		
	Horizontal	0	0	
		0		

Tabla 5.16: Corte en dos dimensiones

Etapa 2

$B = Max\{B_i, i \in I\}$
$B = Max\{260, 138\} = 260$
$B > 0 \Rightarrow P = P \cup \{1\}, P = \{(2,6)\}, n_1 = 0 + 1 = 1, V_T = 56$
Si $(e_1 + e_2) \geq (h_1 + h_2), entonces R = R \cup \{(2,2)\} \cup \{7,8\}$
en caso contrario $R = R \cup \{(7,6)\} \cup \{2,9\}$

Port tanto la pieza cortada es la $(2,6)$ y los nuevos rectángulos a cortar son $(3,6)$ y $(12,6)$

COTA SUPERIOR BK2
Cortamos la pieza $(2,6)$ tantas veces sea posible desde la parte inferior izquierda del tablero. Ahora vamos a analizar la mejor forma de cortar el tablero

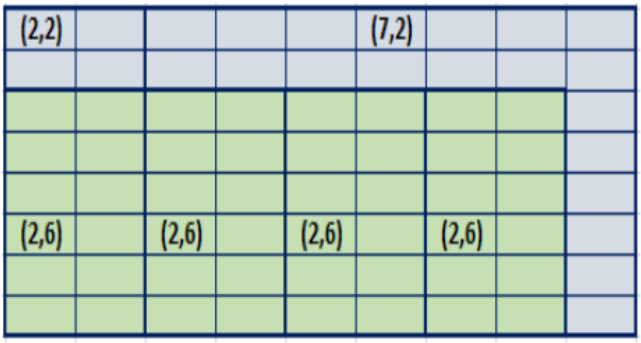


Figura 5.52: Tablero y piezas por cortar

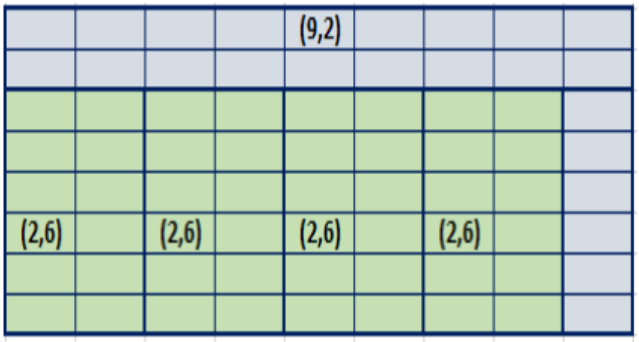


Figura 5.53: Tablero y piezas por cortar

PIEZA 1: Ya no se considera.
PIEZA 2: dimensiones $(7, 3)$: Ya no es posible cortar.

5.6 Bibliografía

- Álvarez-Valdés Ramón, Parajón Antonio, y Tamarit José Manuel. (2007). *GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem*. INFORMS Journal on Computing.

- Álvarez-Valdés Ramón, Parajón Antonio, y Tamarit José Manuel. (2002). *"A Tabu Search Algorithm for Large-scale Guillotine (Un) Constrained Two-dimensional Cutting Problems"*, Computers & Operations Research 29, 925-947.

- Bazaraa, S., Javis, John J. (1996). *Programación lineal y flujo de redes*. México: Limusa. Quinta edición.

- Christofides, N., Whitlock A. (1977). *An algorithm for twodimensional cutting problems. Operational Research*. Vol. 25. pp. 30-44.

- Dyckhoff, H. (1990). *A Typology of Cutting and Packing Problems*. European Journal of Operation Research 44, 145-159.

- Farley, A. (1990). *The cutting stock problem in the canvas industry*. European Journal of Operational Research, 44, 247-255.

- Gilmore, P., Gomory, R. (1961). *A linear programming approach to the cutting stock problem*. Operation Research. 9 849-859.

- Kantorovich, L., Zalgaller V (1951). *Calculation of Rational Cutting of Stock*. Lenizdat, Leningrad.

- Kantorovich, L. (1960). *Mathematical Methods of Organizing and Planning Production*. Management Science 6, No. 4. 363-422.

- Martello, S., Toth, P. (1990). *Knapsack Problems*. Great Britain: John Wiley & Sons.

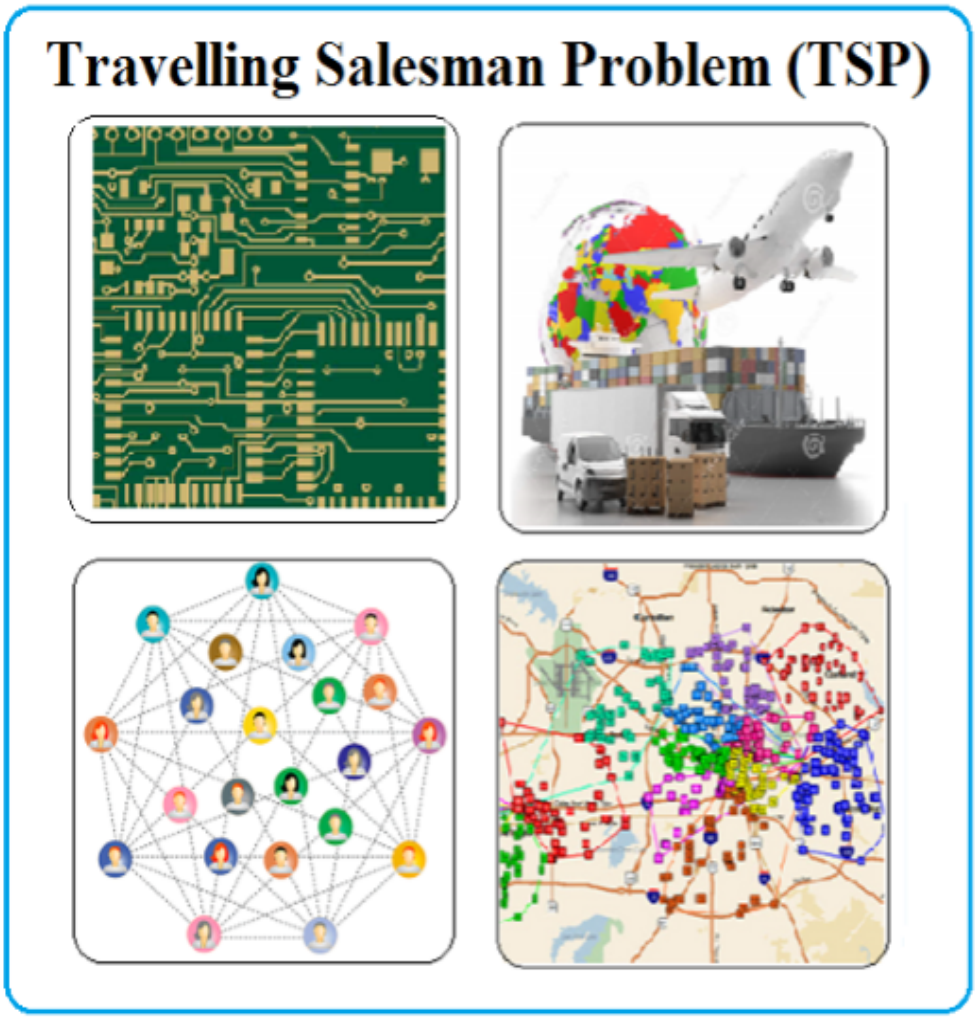
- Mathur, K., Solow D. (1996). *Investigación de Operaciones*. México: Prentice-Hall Hispanoamericana S.A.

- Mobasher, A, Ekici, A. (2012). *Solution approaches for the cutting stock problem with setup cost*. Computers & Operations Research.

- Oliveira, J., Ferreira, J. (1990). *An improved version an improved version of Wang´s algorithm for two - dimensional Cutting Problems*. EJOR 44. pp. 256-266.

- Vasko, F. (1989). *Computational improvement to Wang´s two-dimensional cutting stock algorithm. Computers and Industrial Engineering*. Vol. 16. pp. 109-115.
- Wäscher, G., Haubner, H., Schumann, H. (2007). *An improved typology of cutting and packing problems*. European Journal of Operational Research, 183, 1109-1130.
- Wang, P. (1983). *Two algorithms for constrained twodimensional cutting stock problems*. Operations Research. Vol. 31. 1983. pp. 573-586.
- Yang, C., Sung, T., Weng, W. (2006). *An improved tabu search approach with mixed objective function for one-dimensional cutting stock problems*. Advances in Engineering Software, 37, 502?513.

CAPÍTULO 6
EL PROBLEMA DEL AGENTE VIAJERO



6.1 Introducción

El Problema del Agente Viajero o en inglés Traveling Salesman Problem (TSP), es uno de los problemas más famosos y estudiados a lo largo de la historia en la Investigación de Operaciones, debido a la complejidad de su solución y a la sencillez de su planteamiento, se clasifica dentro de los problemas de Optimización Combinatoria NP-duros. El TSP consiste en que dadas n ciudades y la distancia entre ellas, se debe encontrar la ruta más corta posible de ir desde el punto de partida y visitar cada ciudad una sola vez retornando a la ciudad de origen.

El Problema del Agente Viajero se puede representar por medio de un grafo, en donde los nodos de este, representan cada una de las ciudades y los arcos la distancia existente entre un par de ellas, formando así lo que se conoce como el ciclo o circuito hamiltoniano.

El génesis del TSP es difícil de determinarlo con precisión, sin embargo en el transcurso del tiempo ha sido posible encontrar evidencias de estudios e investigaciones que han sido esenciales en lo que concierne al planteamiento y definición de este problema. Antes de 1800 no se había formulado formalmente, y fue justamente en ese año, que el matemático irlandés William Rowan Hamilton y el matemático británico Thomas Penyngton Kirkamn, elaboran la formulación matemática del problema y además se dio a conocer el juego Icosian de Hamilton, el cual tenía como objetivo dar con el recorrido de Hamilton por las aristas de un dodecaedro para visitar una y sólo una vez cada vértice y que el de llegada coincida con el de partida.

Es generalmente aceptado, por la comunidad científica que la expresión ?Problema del Agente Viajero? se dio a conocer en la Universidad de Princeton entre los años 1931 y 1932. Es justamente en la década de los años 30, particularmente en la Universidad de Harvard, que Merrill Flood tuvo un papel sobresaliente en lo que respecta a la divulgación del problema, ya que fue quien empezó a trabajar en la búsqueda de una ruta óptima para un autobús escolar; de la misma manera en Viena, el matemático Karl Menger enunció lo que se denominada el problema del mensajero planteando muchas de las propiedades del TSP. Considera el algoritmo de fuerza bruta obvio, y observa la falta de optimización de la heurística vecino más cercano.

Ya para década de los 50 y 60 el TSP tiende a popularizarse y de igual manera se inicia la realización de investigaciones con un número mayor de ciudades. En este sentido, G. Dantzig, R. Fulkerson y S. Johnson en el año 1954 proponen una solución al problema, el cual es considerado uno de los principales eventos en la historia de la Optimización Combinatoria, en el cual se resuelve el TSP para

49 ciudades, una por cada estado de los Estados Unidos.

En 1972, Richard M. Karp mostró que el problema del ciclo de Hamilton era NP-completo, lo que implica que el TSP es NP-hard. Esto suministra una explicación matemática de la dificultad computacional aparente de encontrar rutas óptimas. Se hizo un gran avance en la década de 1970 y 1980, cuando Grotschel, Padberg, Rinaldi y otros lograron resolver exactamente los casos con un máximo de $2,392$ ciudades, con planos de corte y la rama y los consolidados. En la década de 1990, Applegate, Bixby, Chvtal, y Cocine desarrollaron el programa Concorde que se ha utilizado en muchas soluciones de registro recientes.

Gerhard Reinelt publicó el TSPLIB que contiene la descripción de una biblioteca para el problema del viajante que está destinada a proporcionar a los investigadores un amplio conjunto de problemas de prueba de diversas fuentes y propiedades, en 1991, es decir, una colección de instancias de referencias de diferentes dificultades, que ha sido utilizado por muchos grupos de investigación para la comparación de resultados, como por ejemplo, en 2006, Cook y otros calculan un recorrido óptimo a través de una instancia de $85,900$ ciudades dada por un problema de diseño microchip. En muchos otros casos, con millones de ciudades, se pueden encontrar soluciones que están garantizados para estar dentro del 1% de un recorrido óptimo.

De ahí en adelante hasta la actualidad se han desarrollado diferentes algoritmos que puedan aplicarse a problemas con un número de ciudades cada vez más grande, gracias al gran desarrollo de la informática y a los estudios que se han venido realizando se han logrado avances importantes en la búsqueda de la mejor solución o la solución óptima para este tipo de problemas.

Como producto de las investigaciones y los estudios referidos al TSP, a lo largo del tiempo se han venido generando diversas variantes del problema, aunque el objetivo fundamental sigue siendo el mismo. Algunas de las variaciones más conocidas se exponen brevemente a continuación:

- Problema del Agente Viajero Generalizado (en inglés, Generalized Traveling Salesman Problem, GTSP). El GTSP divide las ciudades que se deben recorrer en regiones, en donde se debe visitar una sola ciudad por región, buscando que se minimice el costo en el recorrido.
- Problema de Ciclo Simple (en inglés, Simple Cycle Problem, SCP). En esta variación del problema original, se tiene que existen costos para realizar las visitas de las ciudades y hay beneficios por cada una de estas, teniendo en cuenta que no tienen que estar incluidas todas las ciudades en el recorrido.

- Problema del Agente Viajero con Recogida y Entrega de Mercancías (en inglés, Pick-up and Delivery Traveling Salesman Problem, PDTSP). En este problema se realiza el recorrido por las ciudades con un mismo vehículo, el cual posee una capacidad limitada para el transporte de productos, ya que debe proveer o recibir una cantidad determinada de estos en cada una de las ciudades incluidas en el recorrido.
- Problema del Agente Viajero con Ventanas de Tiempo, (en inglés, Traveling Salesman Problem with Time Windows). Este problema posee la característica de manejo del tiempo, en donde para cada ciudad se tiene un tiempo establecido, el cual es mínimo para la llegada y máximo para la salida.
- Problema del Agente Viajero con Múltiples Agentes. Esta variación establece la existencia de un número determinado de agentes, los cuales deben visitar las ciudades, con la limitante de que no podrán visitar aquellas ciudades que ya han sido visitadas por otro de los agentes que hacen parte de la ruta.

El TSP tiene diversas aplicaciones aún en su formulación más simple, tales como la planificación, la logística y en la fabricación de microchips. A continuación, se comentan brevemente algunas de estas aplicaciones:

a. **Logística**: Las aplicaciones más directas y más abundantes del TSP se centran en el campo de la logística. El flujo de personas, mercancías y vehículos en torno a una serie de ciudades o clientes se adapta perfectamente a la filosofía del TSP, como ya demostraron los primeros estudiosos del problema. Entre las múltiples aplicaciones logísticas del problema del viajante, destacamos:

   – **Vendedores y turistas.** Aunque los viajes que se realizan por placer o por negocio rara vez se plantean como un TSP, la mayor parte de los vendedores y turistas utilizan algún planificador de rutas para determinar cuál es el mejor camino para visitar los puntos que desean y volver al punto de origen. Estos planificadores generalmente incluyen algún algoritmo de resolución del TSP.

   – **Rutas escolares.** Las rutas escolares representan una de las primeras aplicaciones del TSP. Actualmente, muchas empresas dedicadas al transporte de personas adquieren software de resolución de TSP que les permite reducir gastos de una manera significativa.

   – **Reparto de correo.** Aunque generalmente el reparto de correo se ajusta mejor a un problema de rutas sobre arcos, en ocasiones el reparto de correo puede modelizarse como un TSP. Se trata de los casos en los que las casas están muy alejadas unas de otras o cuando sólo se debe visitar algunas de ellas (será el caso de las empresas de paquetería). Este esquema es aplicable al reparto de cualquier otro tipo de mercancía.

b. **Industria**: Las aplicaciones en industria no son tan numerosas como en logística, pero la aplicación del problema en este ámbito también ha dado lugar a una significativa reducción de los costos. Entre las aplicaciones a la industria encontramos:

   – **Secuenciación de tareas.** Supongamos que una máquina debe realizar una serie de tareas en el mínimo tiempo posible y sin importar el orden de las mismas. Supongamos que se tarda un tiempo t_{ij} en poner a punto la máquina para realizar la tarea j si la última tarea que realizó fue la i. En ese caso, podemos aplicar un TSP suponiendo que cada tarea es uno de los nodos a visitar, han de realizarse todas las tareas para producir el producto y que la distancia entre ellos es t_{ij}.

   – **Producción de circuitos electrónicos.** La utilización del TSP para la producción de circuitos electrónicos se centra en dos aspectos: el orden óptimo de taladrar las placas y los caminos óptimos necesarios para conectar los chips entre sí.

   – **Problemas de perforado.** Los circuitos integrados se encuentran en muchos dispositivos electrónicos, por lo que la producción de las placas sobre las que se montan dichos circuitos es un problema cotidiano. Dichas placas han de ser perforadas un número relativamente grande de ocasiones. Los orificios resultantes sirven para introducir los chips correspondientes. Generalmente, son taladros automáticos los que realizan, uno tras otro, las perforaciones correspondientes. Si estas máquinas no son programadas correctamente, el tiempo que se tarda en recorrer la placa de un orificio a otro puede aumentar significativamente, dando lugar a pérdidas económicas (si se tarda mucho en producir cada placa, produciremos menos placas en el mismo tiempo). Por tanto, la aplicación del TSP en este campo consiste en, tomando como ciudades cada una de las posiciones donde debe realizarse una perforación y las distancias entre ellas como el tiempo que necesita la máquina en trasladarse de una a otra, minimizar el tiempo que pierde la taladradora en moverse de una posición a otra.

   – **Conexión de chips.** Este tipo de ejemplos se da frecuentemente en el diseño de ordenadores y de otros dispositivos digitales. Dentro de muchos de estos dispositivos existen placas que cuentan con chips que deben ser conectados entre sí por cables. Para evitar problemas de interferencias y debido al pequeño tamaño de los chips, no se pueden poner más de dos cables en un único pin. La idea es, por tanto, minimizar la cantidad de cable necesaria para unir todos los puntos. Claramente este modelo puede ser modelizado como un TSP tomando los pins como las ciudades y la distancia entre ellas, la cantidad de cable necesario para unirlas.

– **Creación de cluster de datos.** La organización de datos en grupos (clusters) de elementos con propiedades similares es un problema básico en análisis de datos. El problema del agente viajero ha sido aplicado frecuentemente en problemas de este tipo cuando existe una buena medida de la similitud $s(a;b)$ entre cada pareja de datos $(a;b)$. La idea es que, usando $s(a;b)$ como distancias, un camino Hamiltoniano de costo máximo situará las observaciones más parecidas cerca unas de otras y se podrá, por tanto, utilizar intervalos del camino como clusters. Cabe destacar que se busca un camino de coste máximo, puesto que la medida de similitud toma un valor mayor cuanto más próximas estén las observaciones entre sí.

Como variantes, aparece como un sub-problema en muchas áreas, como en la secuencia de ADN. El concepto de ?ciudad? puede representar, por ejemplo los clientes, puntos de soldadura o fragmentos de ADN, y el concepto de ?distancia? representa el tiempo de viaje o costo, o una medida de similitud entre los fragmentos de ADN. En muchas aplicaciones, restricciones adicionales como el límite de recursos o el tiempo hacen el problema considerablemente difícil.

Por otra parte, para la resolución del TSP se han empleado técnicas tanto exactas como heurísticas. En lo que respecta a las primeras es posible mencionar al Método de los Planos de Corte, la cual fue una de los primeros métodos utilizados para resolver el Problema del Agente Viajero, está basada en las técnicas de programación lineal, y fue definida por Dantzig en 1954.

Otro procedimiento implementado para resolver el TSP es el método de ramificación y acotación (Branch and Bound). Este se empezó a desarrollar posteriormente a la publicación del método de Planos de Corte. El algoritmo Branch and Bound generalmente es considerado como una extensión de los planos de corte y se utiliza para resolver una gran variedad de problemas.

Así mismo, a principios de los años 60, surgió otra forma de resolver el TSP, en esta ocasión se recurrió a la teoría de programación dinámica definida por Bellman. La idea de este algoritmo es que en un circuito óptimo, tras haber recorrido una serie de ciudades, el camino que atraviese las restantes ciudades debe ser también óptimo, este hecho permite construir el circuito paso a paso.

Otra forma de resolver el problema en cuestión son los métodos heurísticos o aproximados, los cuales generan soluciones razonables, aunque no se pueda evidenciar que sean las soluciones optimas, pero permite que mediante la utilización adecuada de la estructura del problema se pueda obtener una

buena solución; otros métodos son los metaheurísticos, los cuales son híbridos entre heurísticas y van más allá de las soluciones que pueda generar un método heurístico sin combinar.

Los metaheurísticos, conocido como algoritmos genéticos son otras de las técnicas empleadas para la solución del Problema del Agente Viajero, estos mediante la simulación de la selección natural y la adaptación evidenciada en la naturaleza y la implementación de los operadores genéticos (selección, recombinación y mutación) generan nuevas poblaciones de posibles soluciones lo que favorece el proceso y permite que la búsqueda de la solución óptima se genere de manera más precisa, aunque no exacta.

6.2 Formulación matemática del problema

El Problema del Agente Viajero puede plantearse de la siguiente forma: un viajante debe visitar cada una de las ciudades de su zona exactamente una vez y volver a su punto de partida. Conocidos los costos de desplazamiento entre cualquier par de ciudades, ¿cómo debe realizar su itinerario de forma que visite cada ciudad exactamente solo una vez y que es el costo total del circuito sea mínimo?

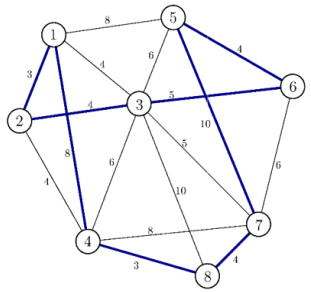


Figura 6.1: Ciclo hamiltoniano

El TSP se puede plantear matemáticamente, como el problema de hallar un ciclo hamiltoniano en un grafo completo en el cual $G = (V, E, C)$, donde se tiene que:

- $V = \{1, 2, 3..., n\}$ es el conjunto de nodos o vértices que representan cada una de las ciudades del problema.
- E, es el conjunto de ramas o aristas.
- C, es la función de costo o distancia, la cual está dada por $C = (c_{ij})$, en donde cada $(i, j) \in E$ y se asigna c_{ij} para asignar el costo o la distancia entre las ciudades i y j. En este trabajo nos referiremos particularmente al TSP simétrico, en el que $c_{ij} = c_{ji}$.

Supongamos que el agente viajero inicia en la ciudad 1 si viaja a las ciudades de i a $i+1$, desde $i=1$, hasta $i=n-1$, y después de la ciudad n a la ciudad 1, esta rutina se puede representar como $1,2,3,?,n+1$ tal orden es conocido como tour. De modo que un tour es un circuito que sale una sola vez de cada ciudad. Por tanto la ciudad inicial es inmaterial y sin pérdida de generalidad podemos decir que se puede elegir cualquier ciudad para que sea la ciudad inicial y de ahí podemos seguir a cualquier otra ciudad $n-1$, así que hay $n-1$ formas distintas de escoger la ciudad que sucede a la primera elección y de esta ciudad se puede viajar a las $n-2$ restantes, etc. De modo que el número total de posibles tours en un problema de n ciudades es de $(n-1)!$

Supongamos que t es un tour dado, se definen las variables enteras como:

$$x_{ij} = \begin{cases} 1 & \text{si viaja de i a j} \\ 0 & \text{De otra forma} \end{cases}$$

Tomando en cuenta las consideraciones antes expuestas, el planteamiento matemático para el TSP es el siguiente:

Función objetivo:

$$Minimizar \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Sujeta a:

$$\sum_{i=1}^{n} x_{ij} = 1 \ \ para\ toda\ j$$
$$\sum_{i=1}^{n} x_{ij} = 1 \ \ para\ toda\ i$$
$$u_i - u_j + (n+1)x_{ij} \leq n \quad para \quad i = 0,1,...,n, \quad j = 0,1,...,n+1; i \neq j$$
$$x_{ij} = \{0,1\} \quad para\ todo \quad i,j$$
$$x = (x_{ij}) \quad es\ un\ tour\ asignado$$

La primer restricción implica que el agente salga sólo una vez de cada ciudad, la segunda a que llegue sólo una vez a cada ciudad, es decir, que la combinación de ambas restricciones asegura que sólo se visite una vez cada ciudad.

La tercer restricción evita la formación de subtour, el conjunto de variables u_i, u_j definen una secuncia de visitas a las ciudades del problema, representando además las ciudades de origen y

destino respectivamente.

Esto implica que si x_{ij} trabajando algebraicamente la tercera restricción tendríamos $u_j \geq u_{i+1}$ es decir, el valor de la variable de llegada debe ser mayor que la variable de partida al menos un una unidad, lo cual impide la formación de sobtours.

Este modelo es general y existen muchas variantes, en particular para evitar la formación de subtours, lo cual genera una gran cantidad de formulaciones y métodos o estrategias de solución en función del tamaño o instancia del problema, es decir, del número de ciudades que debe visitar el agente, anotando que a mayor instancia se incrementa la complejidad de la solución.

6.3 Tratamiento metodológico

El tratamiento metodológico que se realizará con el Problema del Agente Viajero es el siguiente: abordaremos un problema que tiene asociado una instancia de baja dimensión, mediante la aplicación de diferentes métodos de solución. Para iniciar se utilizará el algoritmo de búsqueda exhaustiva, luego implementaramos un método de Programacion Entera, particularmente el algoritmo Branch and Bound, seguidamente se propone encontrar otra solución con un algoritmo heurístico, en este caso, el Vecino Más Cerano y finalmente se confirmarán los resultados obtenidos con la aplicación del WinQsb.

Ejemplo 6.1 *Una empresa nacional tiene sede en las ciudades de Managua* (1)*, León* (2)*, Estelí* (3)*, Matagalpa* (4)*. Cada trimestre del año, uno de los trabajadores del área contable, de la casa matriz ubicada en la ciudad de Managua, tiene que desplazarse a cada una de las cabeceras departamentales y regresar a la ciudad a la capital.*

Figura 6.2: Mapa de las ciudades

Lo que se requiere es determinar la ruta más corta posible, tomando en cuenta que se debe visitar cada ciudad una sola vez y se debe regresar a la ciudad de partida. A continuación se muestran las distancias entre las distintas ciudades:

0	**1**	**2**	**3**	**4**
1	0	93	148	130
2	93	0	131	144
3	148	131	0	70
4	130	144	70	0

Tabla 6.1: *Distancia entre cuidades*

Solución 1: (Implementando el algoritmo exacto de búsqueda exhaustiva)

Recordemos que la búsqueda exhaustiva, es una técnica trivial frecuentemente utilizada para problema con una baja dimensión. La idea fundamental de este algoritmo consiste en enumerar sistemáticamente todos los posibles candidatos para la solución de un problema, con el fin de examinar si dicho candidato satisface la solución al mismo. El modelo matemático (Programación Entera) asociado es el siguiente:

Función objetivo:

Minimizar

$$z = 93x_{12} + 148x_{13} + 130x_{14} + 93x_{21} + 131x_{23} + 144x_{24}$$
$$+148x_{31} + 131x_{32} + 70x_{34} + 130x_{41} + 144x_{42} + 70x_{43}$$

Sujeta a:

$$x_{12} + x_{13} + x_{14} = 1 \quad (1)$$
$$x_{21} + x_{23} + x_{24} = 1 \quad (2)$$
$$x_{31} + x_{32} + x_{34} = 1 \quad (3)$$
$$x_{41} + x_{42} + x_{43} = 1 \quad (4)$$

$$x_{21} + x_{31} + x_{41} = 1 \quad (5)$$
$$x_{12} + x_{32} + x_{42} = 1 \quad (6)$$
$$x_{13} + x_{23} + x_{43} = 1 \quad (7)$$
$$x_{14} + x_{24} + x_{34} = 1 \quad (8)$$

$$x_{ij} \in (1, 0)$$

Con las expresiones (1) hasta (4) se restringe la salida de la ciudad i, y con las expresiones (5) hasta (8) se restringe las llegadas a la ciudad j.

La primera forma para resolver este problema será mediante la enumeración sistemáticamente de cada una de las rutas posibles, y para esto nos auxiliaremos del grafo asociado al problema:

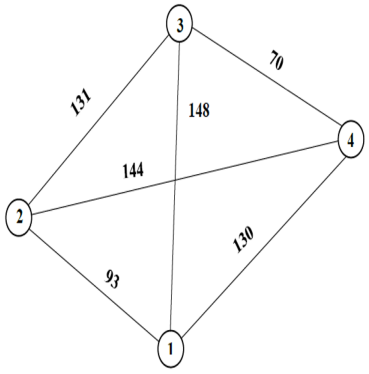


Figura 6.3: Grafo asociado al problema

Partiendo de la ciudad de Managua, los caminos posibles son los siguientes:

1. Managua-León-Matagalpa-Estelí-Managua

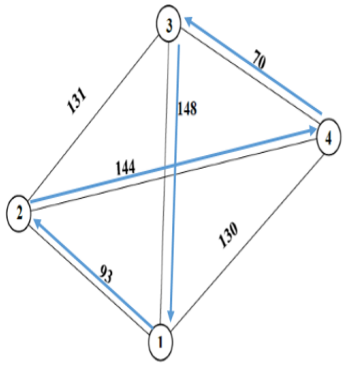


Figura 6.4: circuito 1-2-4-3-1

2. Managua-León-Estelí- Matagalpa-Managua

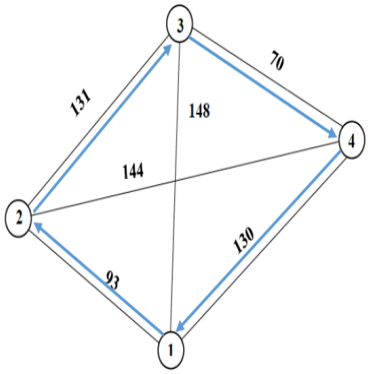


Figura 6.5: circuito 1-2-3-4-1

3. Managua-Matagalpa-León-Estelí-Managua

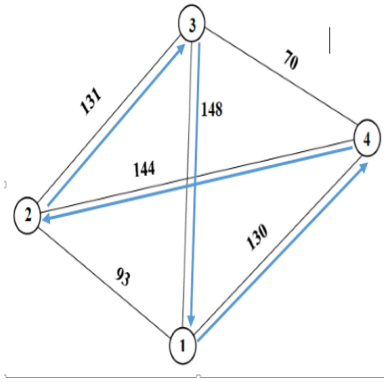


Figura 6.6: circuito 1-4-2-3-1

4. Managua-Estelí-León-Matagalpa-Managua

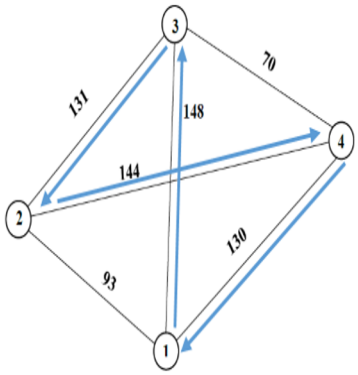


Figura 6.7: circuito 1-3-2-4-1

5. Managua-Matagalpa-Estelí-León-Managua

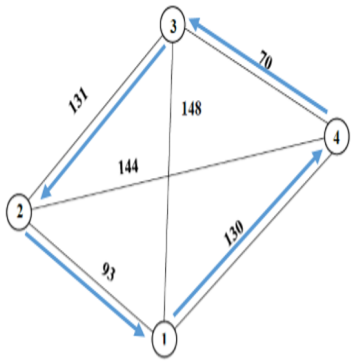


Figura 6.8: circuito 1-4-3-2-1

6. Managua-Estelí-Matagalpa-León-Managua

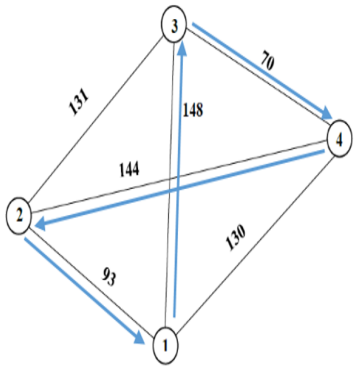


Figura 6.9: circuito 1-3-4-2-1

Mediante la visualización de los diversos nodos podemos apreciar que algunos circuitos son el inverso de otros, por ejemplo: el circuito Managua-León-Matagalpa-Estelí-Managua es el inverso del circuito Managua-Estelí-Matagalpa-León-Managua; a su vez, Managua-León-Estelí-Matagalpa-Managua es el inverso de Managua-Matagalpa-Estelí-León-Managua, y el circuito Managua-Matagalpa-León-Estelí-Managua es el inverso de Managua-Estelí-León-Matagalpa-Managua. Es por ello que para determinar la ruta más corta, únicamente es necesario determinar la distancia en tres de los seis circuitos posibles:

Para el circuito Managua-León-Matagalpa-Estelí-Managua:

1	**2**	**4**	**3**	**1**

Tabla 6.2: *Solución: 93 + 144 + 70 + 148 = 455*

Para el circuito Managua-León-Estelí-Matagalpa-Managua

1	**2**	**3**	**4**	**1**

Tabla 6.3: *Solución: 93 + 131 + 70 + 130 = 424*

Para el circuito Managua-Matagalpa-León-Estelí-Managua

1	**4**	**2**	**3**	**1**

Tabla 6.4: *Solución: 130 + 144 + 131 + 148 = 553*

Comparando las extensiones de los circuitos analizados podemos ver que el más corto es el $1-2-3-4-1$, con una distancia de $424km$, el cual representa el valor óptimo. En consecuencia, el trabajador de la empresa deberá elegir la siguiente ruta: primero debe dirigirse a León, posteriormente a Estelí, de ahí partir a Matagalpa, y finalmente retornar a Managua.

Cabe señalar que con 4 nodos (ciudades) se tendrán $3 \cdot 2 \cdot 1 = 6$ opciones y suprimiendo los tours reversos obtenemos únicamente 3 posibilidades. De la misma manera si tuviéramos 5 nodos, el total de opciones serían $4 \cdot 3 \cdot 2 \cdot 1 = 24$ y al eliminar las trayectorias inversas se tendremos solamente 12 posibilidades. Si seguimos el mismo esquema de razonamiento, para un total de n ciudades tenemos $(n-1)!/2$ posibilidades.

Solución 2: (Implementando el algoritmo exacto Branch and Bound con WinQsb)

El método de Branch and Bound es un algoritmo diseñado para la resolución de modelos de programación entera. Su operatoria consiste en "linealizar" el modelo de Programación Entera, es decir resolver este como si fuese un modelo de Programación Lineal y luego generar cotas en caso que al menos una variable de decisión tome un valor fraccionario. El algoritmo genera en forma recursiva cotas (restricciones adicionales) que favorecen la obtención de valores enteros para las

variables de decisión.

Algoritmo

1. Encontrar la solución mediante el Método Simplex.
2. Iniciar con la solución óptima generada por el Simplex en donde se ignoran las restricciones de variables enteras.
3. Seleccionar una variable no nula y se crean dos ramas mutuamente excluyentes esto da lugar a dos nuevos problemas de Programación Lineal, que se deben resolver.
4. Si ninguna solución es entera, con la rama de mayor valor de la función objetivo, se procede a crear nuevas ramas y se resuelven problemas por Programación Lineal (Método Simplex).
5. Repetir el paso anterior hasta encontrar la solución entera óptima.

La filosofía del algoritmo de Branch and Bound, particularmente para modelos binarios es comenzar con la solución óptima del problema. Si la solución obtenida es un tour, el proceso finaliza, en caso contrario, se procede a establecer restricciones en la solución resultante para impedir la creación de subtours. Luego de esto, la idea es generar ramas con las variables que asignen un valor cero a cada una de las variables de uno de los subtour. Generalmente, el subtour con la menor cantidad de ciudades se elige para la ramificación puesto que es el que crea el menor número de ramas.

Si la solución del problema en cualquier nodo es un tour, su valor objetivo proporciona una cota superior en la longitud óptima del tour. Si no se requiere más ramificación en el nodo. Es importante señalar que un subproblema se explora a fondo si produce una cota superior más pequeña, o si existe evidencia de que no puede conducir a una mejor cota superior. El tour óptimo se encuentra en el nodo con la menor cota superior.

A continuación se expone la solución al problema mediante la aplicación del algoritmo Branch and Bound. Recordemos que el modelo matemático asociado al problema es el siguiente:

Función objetivo:

Minimizar

$$z = 93x_{12} + 148x_{13} + 130x_{14} + 93x_{21} + 131x_{23} + 144x_{24}$$
$$+148x_{31} + 131x_{32} + 70x_{34} + 130x_{41} + 144x_{42} + 70x_{43}$$

Sujeta a:

$$x_{12} + x_{13} + x_{14} = 1 \quad (1)$$
$$x_{21} + x_{23} + x_{24} = 1 \quad (2)$$
$$x_{31} + x_{32} + x_{34} = 1 \quad (3)$$
$$x_{41} + x_{42} + x_{43} = 1 \quad (4)$$
$$x_{21} + x_{31} + x_{41} = 1 \quad (5)$$
$$x_{12} + x_{32} + x_{42} = 1 \quad (6)$$
$$x_{13} + x_{23} + x_{43} = 1 \quad (7)$$
$$x_{14} + x_{24} + x_{34} = 1 \quad (8)$$

$$x_{ij} \in (1, 0)$$

Comenzamos resolviendo el problema inicial, y para ello nos auxiliaremos del WinQsb, y particularmente del módulo referido a Programación Lineal y Entera.

Figura 6.10: Módulo de Programación Lineal y entera en WinQsb

Estando ahí, procedemos a ingresar la información general relacionado con el modelo, de la siguiente forma:

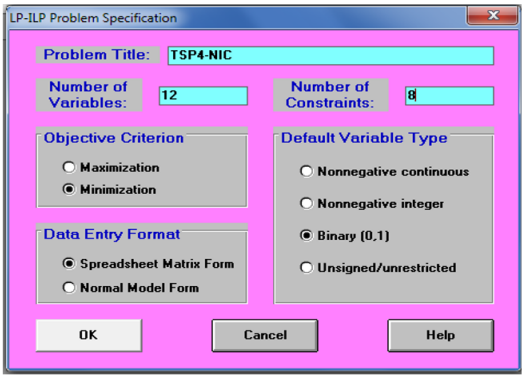


Figura 6.11: Información del problema

Seguidamente ingresamos tanto la función objetivo como las restricciones del modelo, y como consecuencia de esto obtenemos la siguiente matriz:

Variable -->	X12	X13	X14	X21	X23	X24	X31	X32	X34	X41	X42	X43	Direction	R. H. S.
Minimize	93	148	130	93	131	144	148	131	70	130	144	70		
C1	1	1	1	0	0	0	0	0	0	0	0	0	=	1
C2	0	0	0	1	1	1	0	0	0	0	0	0	=	1
C3	0	0	0	0	0	0	1	1	1	0	0	0	=	1
C4	0	0	0	0	0	0	0	0	0	1	1	1	=	1
C5	0	0	0	1	0	0	1	0	0	1	0	0	=	1
C6	1	0	0	0	0	0	0	1	0	0	1	0	=	1
C7	0	1	0	0	1	0	0	0	0	0	0	1	=	1
C8	0	0	1	0	0	1	0	0	1	0	0	0	=	1
LowerBound	0	0	0	0	0	0	0	0	0	0	0	0		
UpperBound	1	1	1	1	1	1	1	1	1	1	1	1		
VariableType	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary	Binary		

Figura 6.12: Datos del problema

Una vez ingresado los datos del modelo, resolvemos y obtenemos la siguiente matriz de salida:

	Decision Variable	Solution Value	Unit Cost or Profit c[j]	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c[j]	Allowable Max. c[j]
1	X12	1.0000	93.0000	93.0000	0	basic	-M	131.0000
2	X13	0	148.0000	0	78.0000	at bound	70.0000	M
3	X14	0	130.0000	0	60.0000	at bound	70.0000	M
4	X21	1.0000	93.0000	93.0000	0	basic	-M	130.0000
5	X23	0	131.0000	0	61.0000	at bound	70.0000	M
6	X24	0	144.0000	0	74.0000	at bound	70.0000	M
7	X31	0	148.0000	0	55.0000	at bound	93.0000	M
8	X32	0	131.0000	0	38.0000	at bound	93.0000	M
9	X34	1.0000	70.0000	70.0000	0	basic	-M	130.0000
10	X41	0	130.0000	0	37.0000	at bound	93.0000	M
11	X42	0	144.0000	0	51.0000	at bound	93.0000	M
12	X43	1.0000	70.0000	70.0000	0	basic	-M	131.0000
	Objective	Function	(Min.) =	326.0000				

Figura 6.13: Solución del problema

La cual nos indica que la solución es $x_{12} = x_{21} = x_{34} = x_{34} = 1$, es decir se forman los subtours $1 - 2 - 1$ y $3 - 4 - 3$ y por tanto procedemos a ramificar, primeramente por x_{12} y luego por x_{21}.

Haciendo $x_{12} = 0$, el modelo quedaría de la siguiente manera:

Función objetivo:

Minimizar

$$z = 148x_{13} + 130x_{14} + 93x_{21} + 131x_{23} + 144x_{24}$$
$$+148x_{31} + 131x_{32} + 70x_{34} + 130x_{41} + 144x_{42} + 70x_{43}$$

Sujeta a:

$$x_{13} + x_{14} = 1 \quad (1)$$
$$x_{21} + x_{23} + x_{24} = 1 \quad (2)$$
$$x_{31} + x_{32} + x_{34} = 1 \quad (3)$$
$$x_{41} + x_{42} + x_{43} = 1 \quad (4)$$

$$x_{21} + x_{31} + x_{41} = 1 \quad (5)$$
$$x_{32} + x_{42} = 1 \quad (6)$$
$$x_{13} + x_{23} + x_{43} = 1 \quad (7)$$
$$x_{14} + x_{24} + x_{34} = 1 \quad (8)$$

$$x_{ij} \in (1, 0)$$

Nuevamente, resolviendo con el WinQsb obtenemos lo siguiente:

	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c(j)	Allowable Max. c(j)
1	X12	0	0	0	0	at bound	0	M
2	X13	0	148.0000	0	31.0000	at bound	117.0000	M
3	X14	1.0000	130.0000	130.0000	0	basic	83.0000	144.0000
4	X21	1.0000	93.0000	93.0000	0	basic	-M	177.0000
5	X23	0	131.0000	0	14.0000	at bound	117.0000	M
6	X24	0	144.0000	0	14.0000	at bound	130.0000	M
7	X31	0	148.0000	0	115.0000	at bound	33.0000	M
8	X32	1.0000	131.0000	131.0000	0	basic	84.0000	145.0000
9	X34	0	70.0000	0	0	basic	56.0000	117.0000
10	X41	0	130.0000	0	84.0000	at bound	46.0000	M
11	X42	0	144.0000	0	0	basic	130.0000	191.0000
12	X43	1.0000	70.0000	70.0000	0	basic	-M	84.0000
	Objective	Function	(Min.) =	424.0000				

Figura 6.14: Solución del subproblema haciendo $x_{12} = 0$

Tenemos que la solución es $x_{14} = x_{21} = x_{32} = x_{43} = 1$, es decir, que con esta ramificación obtenemos como resultado el tour $1 - 4 - 3 - 2 - 1$, obteniendo así un valor $Z = 424$. Esto podría representar la solución óptima, pero previo a realizar dicha afirmación es necesario averiguar qué resultados encontramos en la otra ramificación.

Ahora procedemos a ramificar por $x_{21} = 0$, con lo cual tenemos el modelo Función objetivo:

Minimizar

$$z = 93x_{12}148x_{13} + 130x_{14} + 131x_{23} + 144x_{24}$$
$$+148x_{31} + 131x_{32} + 70x_{34} + 130x_{41} + 144x_{42} + 70x_{43}$$

Sujeta a:

$$x_{12} + x_{13} + x_{14} = 1 \quad (1)$$
$$x_{23} + x_{24} = 1 \quad (2)$$
$$x_{31} + x_{32} + x_{34} = 1 \quad (3)$$
$$x_{41} + x_{42} + x_{43} = 1 \quad (4)$$
$$x_{31} + x_{41} = 1 \quad (5)$$
$$x_{12} + x_{32} + x_{42} = 1 \quad (6)$$
$$x_{13} + x_{23} + x_{43} = 1 \quad (7)$$
$$x_{14} + x_{24} + x_{34} = 1 \quad (8)$$
$$x_{ij} \in (1, 0)$$

Resolviendo de forma análoga tenemos que

	Decision Variable	Solution Value	Unit Cost or Profit c[j]	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c[j]	Allowable Max. c[j]
1	X12	1.0000	93.0000	93.0000	0	basic	-M	174.0000
2	X13	0	148.0000	0	17.0000	at bound	131.0000	M
3	X14	0	130.0000	0	17.0000	at bound	113.0000	M
4	X21	0	0	0	0	at bound	0	M
5	X23	1.0000	131.0000	131.0000	0	basic	88.0000	148.0000
6	X24	0	144.0000	0	31.0000	at bound	113.0000	M
7	X31	0	148.0000	0	0	basic	131.0000	191.0000
8	X32	0	131.0000	0	81.0000	at bound	50.0000	M
9	X34	1.0000	70.0000	70.0000	0	basic	-M	87.0000
10	X41	1.0000	130.0000	130.0000	0	basic	87.0000	147.0000
11	X42	0	144.0000	0	112.0000	at bound	32.0000	M
12	X43	0	70.0000	0	0	basic	53.0000	113.0000
	Objective	Function	(Min.) =	424.0000				

Figura 6.15: Solución del subproblema haciendo $x_{21} = 0$

En esta ocasión tenemos como solución $x_{12} = x_{23} = x_{34} = x_{41} = 1$, es decir, que con esta ramificación obtenemos como resultado el tour $1 - 4 - 3 - 2 - 1$, el cual resulta ser el tour inverso al obtenido en la ramificación anterior. Y por supuesto tenemos un valor $Z = 424$. Como ya hemos encontrado un tour completo, entonces podemos afirmar que tenemos la solución óptima, la cual es $1 - 4 - 3 - 2 - 1$, con un valor $Z = 424$. Haciendo un resumen gráfico del proceso implementado por Branch and Bound, tenemos:

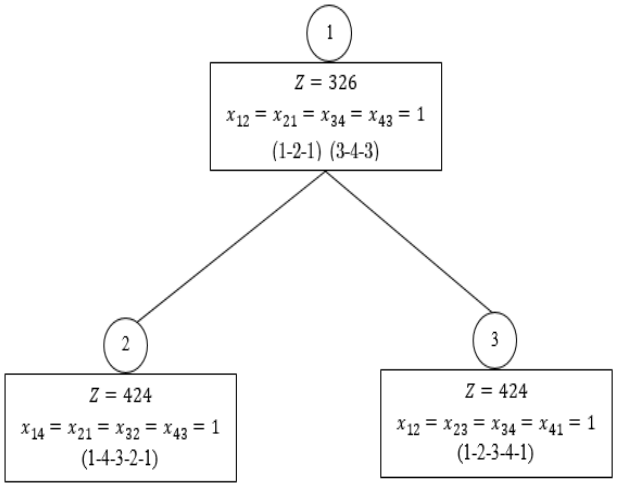


Figura 6.16: Ramificación y acotamiento

Con todo esto hemos podido corroborar que ya sea usando el algoritmo de Fuerza Bruta o bien el método Branch and Bound, llegamos a la misma solución. Claro está que el tiempo que toma implementar el primer método es mucho mayor que el del segundo algoritmo.

Solución 3: (Implementando el algoritmo heurístico Vecino más Cercano)

Ahora aplicaremos un algoritmo heurístico conocido como El Vecino más Cercano, el cual fue uno de los primeros procedimientos implementados para resolver el TSP. Mediante la ejecución del algoritmo es posible encontrar rápidamente un camino corto, pero generalmente no el ideal.

Abajo está la aplicación del algoritmo del vecino más próximo al problema del viajante. Estos son los pasos del algoritmo:

Paso 1: Elegir un nodo arbitrario respecto al nodo actual.
Paso 2: Determinar la arista de menor peso que esté conectada al nodo actual y a un nodo no visitado N.
Paso 3: Convertir el nodo actual en N.
Paso 4: Marcar N como visitado.
Paso 5: Si todos los nodos ya estuvieran visitados, cerrar el algoritmo.
Paso 6: Vaya al paso 2.

El algoritmo del Vecino más Cercano es fácil de implementar y ejecutar rápidamente, sin embargo, algunas veces puede perder tours más cortos, que con la vista humana son fácilmente perceptibles, esto ocurre debido su naturaleza de algoritmo voráz. Como regla general, si los últimos pasos del recorrido son comparables en longitud al de los primeros pasos, el recorrido es razonable; si estos son mucho mayores, entonces es probable que existan caminos mucho mejores.

El pseudocódigo es el siguiente:

Algoritmo del Vecino más Cercano
Inicialización Seleccionar un vértice j al azar. Hacer $t = j$ y $W = V$ / $\{j\}$. Mientras $(W \neq \phi)$ Tomar $j \in W / c_{tj} = min\{c_{ti} / i \in W\}$ Conectar t a j Hacer $W = W\ j$ y $t = j$.

Tabla 6.5: *Pseudocódigo del algoritmo de Dijkstra*

Al implementar el algoritmo obtenemos lo siguiente:

Inicializa en el nodo 1.

$j = 1$
$t = 1$ y $W = \{1, 2, 3, 4\}/\{1\}$
$W = \{2, 3, 4\}$

Mientras $W \neq \phi$
Tomar $j \in W / c_{1j} = min\{c_{12}, c_{13}, c_{14}\}$
$= min\{93, 148, 130\}$
$= 93 = \mathbf{c_{12}}$

Conectar el nodo 1 con el nodo 2
Hacer $W = \{3, 4\}$ y $t = 2$

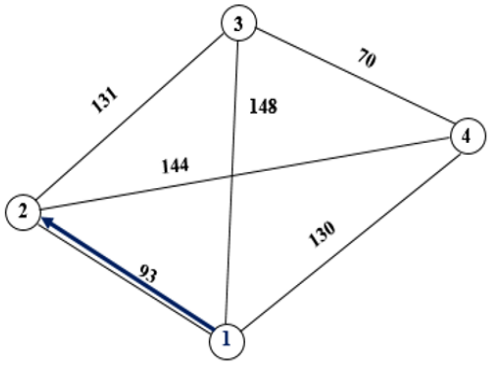


Figura 6.17: conexión de nodo 1 y nodo 2

Mientras $W \neq \phi$
Tomar $j \in W / c_{2j} = min\{c_{23}, c_{24}\}$
$= min\{131, 144\}$
$= 131$
$= \mathbf{c_{23}}$

Conectar el nodo 2 con el nodo 3
Hacer $W = \{4\}$ y $t = 3$

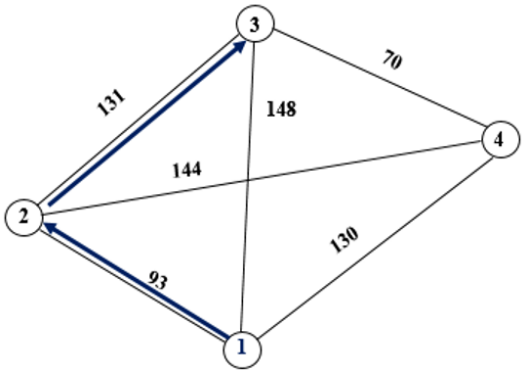


Figura 6.18: conexión de nodo 2 y nodo 3

Mientras $W \neq \phi$
Tomar $j \in W / c_{3j} = min\{c_{34}\}$
$= min\{70\}$
$= 70$
$= \mathbf{c_{34}}$

Conectar el nodo 3 con el nodo 4
Hacer $W = \{\quad\}$ y $t = 4$

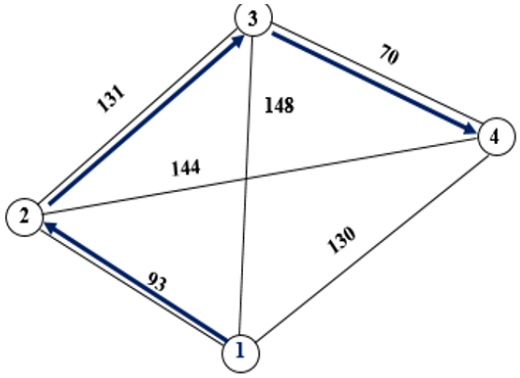


Figura 6.19: conexión de nodo 3 y nodo 4

Como $W = \phi$ se detiene el algoritmo y se procede a unir el nodo 4 con el nodo 1 y calculamos la solución:

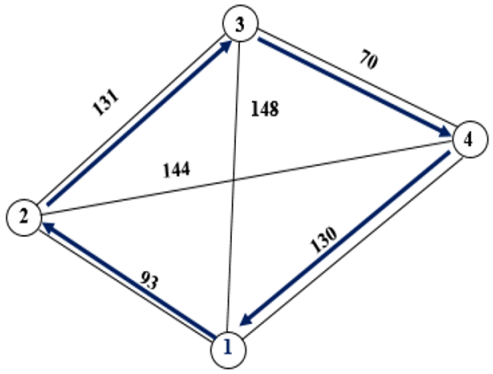


Figura 6.20: conexión de nodo 4 y nodo 1

Por tanto la solución es:

1	**2**	**3**	**4**	**1**

Tabla 6.6: *Solución: 93 + 131 + 70 + 130 = 424*

Con la antes expuesto, es posible apreciar que la gran ventaja del Vecino más Cercano es que entrega una solución razonable (en este caso exacta) en mucho menos tiempo que el método de fuerza bruta,

es decir que no fue necesario la exploración de todos el espacio de soluciones para encontrar la solución óptima.

Solución 4:(Implementando la herramienta computacional WinQsb)

A continuación resolveremos el mismo problema haciendo uso de la herramienta computacional WinQsb. La idea fundamental es la comprobación de los resultados encontrados mediante la ejecución de los distintos algoritmos previamente expuestos.

a. Elegir módulo de modelado de red.

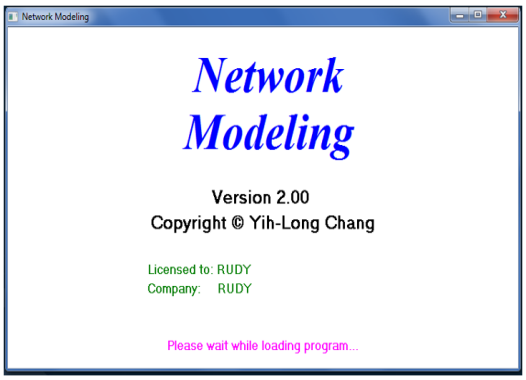


Figura 6.21: Módulo de Modelo de Redes del WinQsb

b. Seleccionar nuevo problema y Traveling Salesman Problem

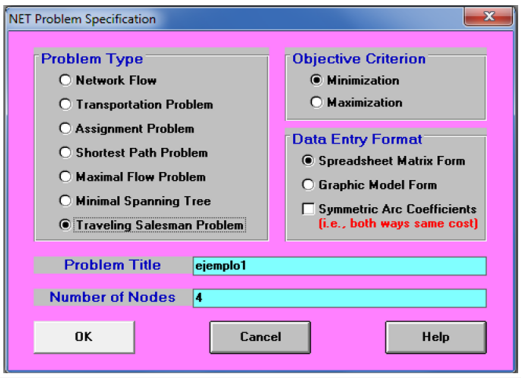


Figura 6.22: Especificaciones del problema de red para el TSP

c. Ingresar los datos del problema

From \ To	Managua	León	Estelí	Matagalpa
Managua	0	93	148	130
León	93	0	131	144
Estelí	148	131	0	70
Matagalpa	130	144	70	0

Figura 6.23: Instancia del TSP

d. Elegir el algoritmo a implementar para resolver el problema

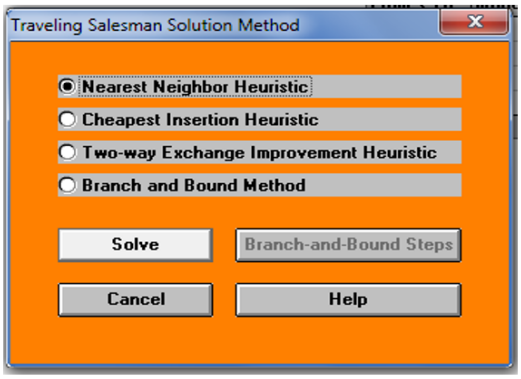


Figura 6.24: Algoritmos del WinQsb para resolver el TSP

e. Mostrar la solución del problema
Con el Vecino más Cercano obtenemos:

From Node	Connect To	Distance/Cost		From Node	Connect To	Distance/Cost
Node1	Node2	93	3	Node3	Node4	70
Node2	Node3	131	4	Node4	Node1	130
Total	Minimal	Traveling	Distance	or Cost	=	424
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.25: Solución del TSP con el vecino más cercano

Con Branch and Bound tenemos:

From Node	Connect To	Distance/Cost		From Node	Connect To	Distance/Cost
Node1	Node4	130	3	Node3	Node2	131
Node4	Node3	70	4	Node2	Node1	93
Total	Minimal	Traveling	Distance	or Cost	=	424
(Result	from	Branch	and	Bound	Method)	

Figura 6.26: Solución del TSP con Con Branch and Bound

De las tablas anteriores podemos observar que con el algoritmo del Vecino Más Cercano, así como el de Branch and Bound llegamos a obtener un valor óptimo igual a $\mathbf{424}$. Además de lo antes expuesto, fue posible confirmar que se obtuvimos resultados exactos con cada uno de los algoritmos implementados.

6.4 Problemas propuestos

1. La empresa de envíos por correo ZOOM desea hacer entrega de cartas y paquetes en las ciudades 1, 2, 3 y 4. Uno de los mensajeros está situado en la ciudad 1 y debe recorrer cada uno de los lugares y regresar al lugar de origen, de tal manera que cada sitio se visite solo una vez y que la distancia recorrida sea mínima. Determinar el cual debe ser la trayectoria a elegir. La cantidad de kilómetros de recorrido entre las ciudades que aparecen en la siguiente tabla.

Ciudad	**1**	**2**	**3**	**4**
1	0	9	7	8
2	9	0	10	15
3	7	10	0	4
4	8	15	4	0

Tabla 6.7: *Instancia del problema* 1

2. Una empresa de producción tiene 5 filiales establecidas en una región y debe proveer a cada uno con los materiales necesarios para su funcionamiento. Desde su sede central o depósito, 1 se encarga de la logística en el cumplimiento de las demandas. Para ello con el único camión que posee esta empresa en construcción debe visitar cada una de las sedes y regresar de nuevo al depósito. Como tiene varias alternativas para recorrer las filiales, se requiere encontrar en qué orden debe hacer el recorrido para minimizar el costo total del trayecto recorrido. En la siguiente tabla se muestran las dependencias, las rutas existentes y el costo de ir de una a otra:

Ciudad	**1**	**2**	**3**	**4**	**5**
1	0	10	6	8	7
2	10	0	20	15	16
3	6	20	0	7	8
4	8	15	7	0	12
5	7	16	8	12	0

Tabla 6.8: *Instancia del problema* 2

3. Una persona vive en la ciudad 1 y tiene 5 agencias en diferentes lugares y desea visitarlos, la distancia entre cada agencia se muestra en la tabla, en qué orden tendrá que visitar las agencias para minimizar la distancia recorrida

Ciudad	**1**	**2**	**3**	**4**	**5**
1	0	132	217	164	58
2	132	0	290	201	79
3	217	290	0	113	303
4	164	201	113	0	196
5	58	79	303	196	0

Tabla 6.9: *Instancia del problema* 3

4. El notificador de la fiscalía vive en la zona 1, parte de su casa y debe recorrer las distintas zonas de la ciudad: 1, 2, 3, 4, 5 y 6 para entregar las notificaciones que se le han encomendado. Finalmente, debe regresar a su casa. Como tiene varias alternativas para recorrer los distintos puntos, se requiere encontrar en qué orden debe hacer el recorrido. La matriz de distancias es la siguiente:

Zona	**1**	**2**	**3**	**4**	**5**	**6**
1	0	8	10	3	5	4
2	8	0	1	5	9	12
3	10	1	0	7	2	21
4	3	5	7	0	6	8
5	5	9	2	6	0	24
6	4	12	21	8	24	0

Tabla 6.10: *Instancia del problema* 4

5. Un repartidor de pizza necesita entregar 7 pedidos en diferentes colonias de la ciudad, 1, 2, 3, 4, 5, 6, 7. El problema consiste en determinar el recorrido que tiene que hacer el repartidor de pizza de tal forma que la distancia recorrida sea la mínima.

Colonia	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	10	12	19	23	22	12
2	10	0	12	16	24	20	22
3	12	12	0	9	18	17	25
4	19	16	9	0	10	17	21
5	23	24	18	10	0	8	20
6	22	20	17	17	8	0	10
7	12	22	25	21	20	10	0

Tabla 6.11: *Instancia del problema* 5

6.5 Problemas resueltos

Problema 1
Solución 1: (Implementando el heurístico del Vecino más Cercano)

Ciudad	**1**	**2**	**3**	**4**
1	0	9	7	8
2	9	0	10	15
3	7	10	0	4
4	8	15	4	0

Tabla 6.12: *Instancia del problema* 1

Inicializa en el nodo 1.
$j = 1$
$t = 1$ y $W = \{1, 2, 3, 4\}/\{1\}$
$W = \{2, 3, 4\}$

Mientras $W \neq \phi$
Tomar $j \in W / c_{1j} = min\{c_{12}, c_{13}, c_{14}\}$

$= min\{9, 7, 8\}$
$= 7 = c_{\mathbf{13}}$

Conectar el nodo 1 con el nodo 3
Hacer $W = \{2, 4\}$ y $t = 3$

Mientras $W \neq \phi$
Tomar $j \in W / c_{3j} = min\{c_{32}, c_{34}\}$
$= min\{10, 4\}$
$= 4 = c_{\mathbf{34}}$

Conectar el nodo 3 con el nodo 4
Hacer $W = \{2\}$ y $t = 4$

Mientras $W \neq \phi$
Tomar $j \in W / c_{4j} = min\{c_{42}\}$
$= min\{15\}$
$= 15 = c_{\mathbf{34}}$

Conectar el nodo 4 con el nodo 2
Hacer $W = \{\}$ y $t = 2$

Como $W \neq \phi$ se detiene el algoritmo y se procede a unir el nodo 2 con el nodo 1 y calculamos la solución.

Por tanto la solución es:

1	**3**	**4**	**2**	**1**

Tabla 6.13: *Solución:7+4+15+9=35*

Solución2: (Implementando el WinQsb)

a. Ingresando los datos del problema

From \ To	Node1	Node2	Node3	Node4
Node1		9	7	8
Node2	9		10	15
Node3	7	10		4
Node4	8	15	4	

Figura 6.27: Datos del problema

b. Eligiendo el algoritmo a implementar:

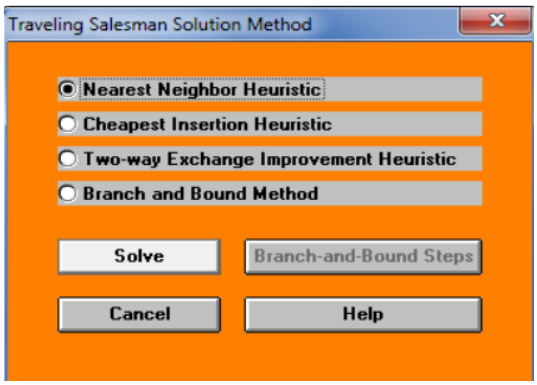


Figura 6.28: Algoritmos del WinQsb para resolver el TSP

c. Mostrando la solución del problema

From Node	Connect To	Distance/Cost		From Node	Connect To	Distance/Cost
Node1	Node3	7	3	Node4	Node2	15
Node3	Node4	4	4	Node2	Node1	9
Total	Minimal	Traveling	Distance	or Cost	=	35
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.29: Solución del TSP con WinQsb

Problema 2.
Solución1: (Implementando el heurístico del Vecino más Cercano)

La matriz asociada al problema es la siguiente:

Ciudad	**1**	**2**	**3**	**4**	**5**
1	0	10	6	8	7
2	10	0	20	15	16
3	6	20	0	7	8
4	8	15	7	0	12
5	7	16	8	12	0

Tabla 6.14: *Instancia del problema* 2

Al implementar el algoritmo obtenemos lo siguiente: **Inicializa en el nodo** 1.
$j = 1$
$t = 1$ y $W = \{1, 2, 3, 4, 5\}/\{1\}$
$W = \{2, 3, 4, 5\}$

Mientras $W \neq \phi$
Tomar $j \in W/c_{1j} = min\{c_{12}, c_{13}, c_{14}, c_{15}\}$
$= min\{10, 6, 8, 7\}$
$= 6 = c_{\mathbf{13}}$

Conectar el nodo 1 con el nodo 3
Hacer $W = \{2, 4, 5\}$ y $t = 3$

Mientras $W \neq \phi$
Tomar $j \in W/c_{3j} = min\{c_{32}, c_{34}, c_{35}\}$
$= min\{20, 7, 8\}$
$= 4 = c_{\mathbf{34}}$

Conectar el nodo 3 con el nodo 4
Hacer $W = \{2, 5\}$ y $t = 4$

Mientras $W \neq \phi$

Tomar $j \in W / c_{4j} = min\{c_{42}, c_{45}\}$
$= min\{15, 12\}$
$= 12 = c_{\mathbf{45}}$

Conectar el nodo 4 con el nodo 5
Hacer $W = \{2\}$ y $t = 5$

Mientras $W \neq \phi$
Tomar $j \in W / c_{5j} = min\{c_{52}\}$
$= min\{16\}$
$= 12 = c_{\mathbf{52}}$
Conectar el nodo 5 con el nodo 2
Hacer $W = \{0\}$ y $t = 2$

Como $W \neq \phi$ se detiene el algoritmo y se procede a unir el nodo 2 con el nodo 1 y calculamos la solución

Por tanto la solución es:

1	**3**	**4**	**5**	**2**	**1**

Tabla 6.15: *Solución:6+7+12+16+10=51*

Solución2: (Implementando el WinQsb)

a. Ingresando los datos del problema

From \ To	Node1	Node2	Node3	Node4	Node5
Node1		10	6	8	7
Node2	10		20	15	16
Node3	6	20		7	8
Node4	8	15	7		12
Node5	7	16	8	12	

Figura 6.30: Datos del problema

b. Eligiendo el algoritmo a implementar:

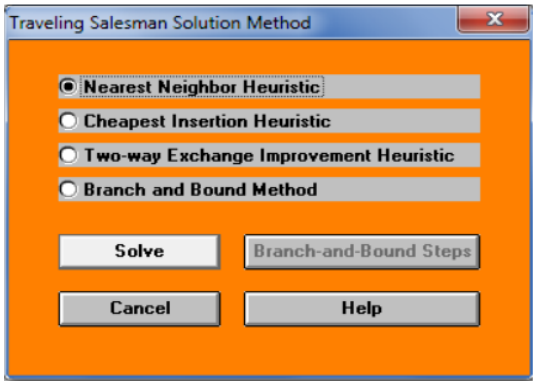


Figura 6.31: Algoritmos del WinQsb para resolver el TSP

c. Mostrando la solución del problema

From Node	Connect To	istance/Cos		From Node	Connect To	Distance/Cost
Node1	Node3	6	4	Node5	Node2	16
Node3	Node4	7	5	Node2	Node1	10
Node4	Node5	12				
Total	Minimal	Traveling	Distance	or Cost	=	51
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.32: Solución del TSP con WinQsb

Problema 3.
Solución1: (Implementando el heurístico del Vecino más Cercano)

La matriz asociada al problema es la siguiente:

Ciudad	**1**	**2**	**3**	**4**	**5**
1	0	132	217	164	58
2	132	0	290	201	79
3	217	290	0	113	303
4	164	201	113	0	196
5	58	79	303	196	0

Tabla 6.16: *Instancia del problema* 3

Al implementar el algoritmo obtenemos lo siguiente: **Inicializa en el nodo** 1.
$j = 1$
$t = 1$ y $W = \{1, 2, 3, 4, 5\}/\{1\}$
$W = \{2, 3, 4, 5\}$

Mientras $W \neq \phi$
Tomar $j \in W / c_{1j} = min\{c_{12}, c_{13}, c_{14}, c_{15}\}$
$= min\{132, 217, 164, 58\}$
$= 58 = \mathbf{c_{15}}$

Conectar el nodo 1 con el nodo 5
Hacer $W = \{2, 3, 4\}$ y $t = 5$

Mientras $W \neq \phi$
Tomar $j \in W / c_{5j} = min\{c_{52}, c_{53}, c_{54}\}$
$= min\{79\}$
$= 4 = \mathbf{c_{52}}$

Conectar el nodo 5 con el nodo 2
Hacer $W = \{3, 4\}$ y $t = 2$

Mientras $W \neq \phi$
Tomar $j \in W / c_{2j} = min\{c_{23}, c_{24}\}$
$= min\{290, 201\}$
$= 201 = \mathbf{c_{24}}$

Conectar el nodo 2 con el nodo 4
Hacer $W = \{3\}$ y $t = 4$

Mientras $W \neq \phi$
Tomar $j \in W / c_{4j} = min\{c_{43}\}$
$= min\{16\}$
$= 113 = \mathbf{c_{43}}$
Conectar el nodo 4 con el nodo 3
Hacer $W = \{0\}$ y $t = 3$

Como $W \neq \phi$ se detiene el algoritmo y se procede a unir el nodo 3 con el nodo 1 y calculamos la solución

Por tanto la solución es:

1	**5**	**2**	**4**	**3**	**1**

Tabla 6.17: *Solución:58+79+201+113+217=668*

Solución2: (Implementando el WinQsb)

a. Ingresando los datos del problema

From \ To	Node1	Node2	Node3	Node4	Node5
Node1		132	217	164	58
Node2	132		290	201	79
Node3	217	290		113	303
Node4	164	201	113		196
Node5	58	79	303	196	

Figura 6.33: Datos del problema

b. Eligiendo el algoritmo a implementar:

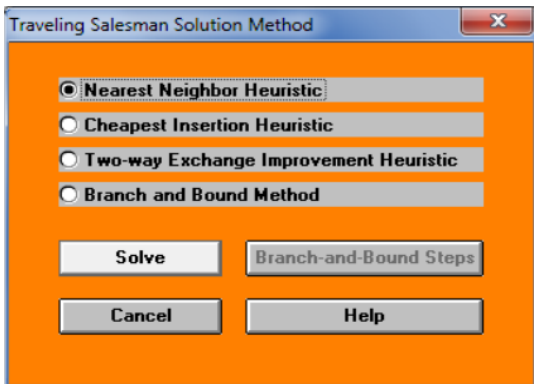


Figura 6.34: Algoritmos del WinQsb para resolver el TSP

c. Mostrando la solución del problema

From	Connect	istance/Co		From	Connect	istance/Co
Node1	Node5	58	4	Node4	Node3	113
Node5	Node2	79	5	Node3	Node1	217
Node2	Node4	201				
Total	Minimal	Traveling	Distance	or Cost	=	668
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.35: Solución del TSP con WinQsb

Problema 5.
Solución1: (Implementando el heurístico del Vecino más Cercano)

La matriz asociada al problema es la siguiente:

Zona	**1**	**2**	**3**	**4**	**5**	**6**
1	0	8	10	3	5	4
2	8	0	1	5	9	12
3	10	1	0	7	2	21
4	3	5	7	0	6	8
5	5	9	2	6	0	24
6	4	12	21	8	24	0

Tabla 6.18: *Instancia del problema* 5

Al implementar el algoritmo obtenemos lo siguiente: **Inicializa en el nodo** 1.
$j = 1$
$t = 1$ y $W = \{1, 2, 3, 4, 5\}/\{1\}$
$W = \{2, 3, 4, 5, 6\}$

Mientras $W \neq \phi$
Tomar $j \in W/c_{1j} = min\{c_{12}, c_{13}, c_{14}, c_{15}, c_{16}\}$
$= min\{8, 10, 3, 5, 4\}$
$= 3 = c_{\mathbf{14}}$

Conectar el nodo 1 con el nodo 4
Hacer $W = \{2, 3, 5, 6\}$ y $t = 4$

Mientras $W \neq \phi$
Tomar $j \in W / c_{4j} = min\{c_{42}, c_{43}, c_{45}, c_{46}\}$
$= min\{5, 7, 6, 8\}$
$= 5 = \boldsymbol{c_{42}}$

Conectar el nodo 4 con el nodo 2
Hacer $W = \{3, 5, 6\}$ y $t = 2$

Mientras $W \neq \phi$
Tomar $j \in W / c_{2j} = min\{c_{23}, c_{25}, c_{26}\}$
$= min\{1, 9, 12\}$
$= 1 = \boldsymbol{c_{23}}$

Conectar el nodo 2 con el nodo 3
Hacer $W = \{5, 6\}$ y $t = 3$

Mientras $W \neq \phi$
Tomar $j \in W / c_{3j} = min\{c_{35}, c_{36}\}$
$= min\{2, 21\}$
$= 2 = \boldsymbol{c_{35}}$

Conectar el nodo 3 con el nodo 5
Hacer $W = \{6\}$ y $t = 5$

Mientras $W \neq \phi$
Tomar $j \in W / c_{5j} = min\{c_{56}\}$
$= min\{24\}$
$= 2 = \boldsymbol{c_{56}}$

Conectar el nodo 5 con el nodo 6
Hacer $W = \{0\}$ y $t = 6$

Como $W \neq \phi$ se detiene el algoritmo y se procede a unir el nodo 6 con el nodo 1 y calculamos la solución

Por tanto la solución es:

1	**4**	**2**	**3**	**5**	**6**	**1**

Tabla 6.19: *Solución:3+5+1+2+24+4=39*

Solución2: (Implementando el WinQsb)

a. Ingresando los datos del problema

From \	Node1	Node2	Node3	Node4	Node5	Node6
Node1		8	10	3	5	4
Node2	8		1	5	9	12
Node3	10	1		7	2	21
Node4	3	5	7		6	8
Node5	5	9	2	6		24
Node6	4	12	21	8	24	

Figura 6.36: Datos del problema

b. Eligiendo el algoritmo a implementar:

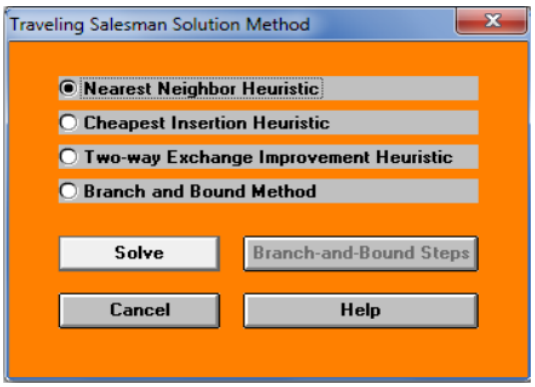


Figura 6.37: Algoritmos del WinQsb para resolver el TSP

c. Mostrando la solución del problema

From	Connect	stance/Co		From	Connect	stance/Co
Node1	Node4	3	4	Node3	Node5	2
Node4	Node2	5	5	Node5	Node6	24
Node2	Node3	1	6	Node6	Node1	4
Total	Minimal	Traveling	Distance	or Cost	=	39
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.38: Solución del TSP con WinQsb

Problema 5.
Solución1: (Implementando el heurístico del Vecino más Cercano)

La matriz asociada al problema es la siguiente:

Colonia	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	10	12	19	23	22	12
2	10	0	12	16	24	20	22
3	12	12	0	9	18	17	25
4	19	16	9	0	10	17	21
5	23	24	18	10	0	8	20
6	22	20	17	17	8	0	10
7	12	22	25	21	20	10	0

Tabla 6.20: *Instancia del problema* 5

Al implementar el algoritmo obtenemos lo siguiente: **Inicializa en el nodo** 1.
$j = 1$
$t = 1$ y $W = \{1,2,3,4,5,6,7\}/\{1\}$
$W = \{2,3,4,5,6,7\}$

Mientras $W \neq \phi$
Tomar $j \in W/c_{1j} = min\{c_{12}, c_{13}, c_{14}, c_{15}, c_{16}, c_{17}\}$
$= min\{10, 12, 19, 23, 22, 12\}$
$= 10 = c_{\mathbf{12}}$

Conectar el nodo 1 con el nodo 2
Hacer $W = \{3, 4, 5, 6, 7\}$ y $t = 2$

Mientras $W \neq \phi$
Tomar $j \in W / c_{2j} = min\{c_{23}, c_{24}, c_{25}, c_{26}, c_{27}\}$
$= min\{12, 16, 24, 20, 22\}$
$= 12 = \mathbf{c_{23}}$

Conectar el nodo 2 con el nodo 3
Hacer $W = \{4, 5, 6, 7\}$ y $t = 3$

Mientras $W \neq \phi$
Tomar $j \in W / c_{3j} = min\{c_{23}, c_{25}, c_{26}\}$
$= min\{1, 9, 12\}$
$= 1 = \mathbf{c_{23}}$

Conectar el nodo 2 con el nodo 3
Hacer $W = \{4, 5, 6, 7\}$ y $t = 3$

Mientras $W \neq \phi$
Tomar $j \in W / c_{3j} = min\{c_{34}, c_{35}, c_{36}, c_{37}\}$
$= min\{9, 18, 17, 25\}$
$= 9 = \mathbf{c_{34}}$

Conectar el nodo 3 con el nodo 4
Hacer $W = \{5, 6, 7\}$ y $t = 4$

Mientras $W \neq \phi$
Tomar $j \in W / c_{4j} = min\{c_{45}, c_{46}, c_{47}\}$
$= min\{10, 17, 21\}$
$= 10 = \mathbf{c_{45}}$

Conectar el nodo 4 con el nodo 5
Hacer $W = \{6, 7\}$ y $t = 5$

Mientras $W \neq \phi$
Tomar $j \in W / c_{5j} = min\{c_{56}, c_{57}\}$
$= min\{8, 20\}$
$= 8 = \mathbf{c_{56}}$

Conectar el nodo 5 con el nodo 6
Hacer $W = \{7\}$ y $t = 6$

Mientras $W \neq \phi$
Tomar $j \in W / c_{6j} = min\{c_{67}\}$
$= min\{10\}$
$= 10 = \mathbf{c_{67}}$

Conectar el nodo 6 con el nodo 7
Hacer $W = \{\}$ y $t = 7$

Como $W \neq \phi$ se detiene el algoritmo y se procede a unir el nodo 7 con el nodo 1 y calculamos la solución

Por tanto la solución es:

1	**2**	**3**	**4**	**5**	**6**	**7**	**1**

Tabla 6.21: *Solución:10+12+9+10+8+10+12=71*

Solución2: (con el WinQsb)

a. Ingresando los datos del problema

From \	Node1	Node2	Node3	Node4	Node5	Node6	Node7
Node1		10	12	19	23	22	12
Node2	10		12	16	24	20	22
Node3	12	12		9	18	17	25
Node4	19	16	9		10	17	21
Node5	23	24	18	10		8	20
Node6	22	20	17	17	8		10
Node7	12	22	25	21	20	10	

Figura 6.39: Datos del problema

b. Eligiendo el algoritmo a implementar:

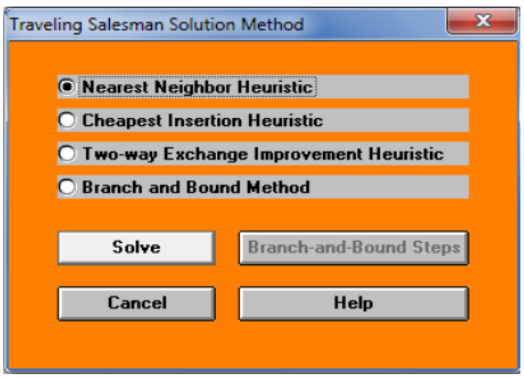


Figura 6.40: Algoritmos del WinQsb para resolver el TSP

c. Mostrando la solución del problema

From Node	Connect	istance/Co		From Node	Connect	istance/Co
Node1	Node2	10	5	Node5	Node6	8
Node2	Node3	12	6	Node6	Node7	10
Node3	Node4	9	7	Node7	Node1	12
Node4	Node5	10				
Total	Minimal	Traveling	Distance	or Cost	=	71
(Result	from	Nearest	Neighbor	Heuristic)		

Figura 6.41: Solución del TSP con el WinQsb

6.6 Bibliografía

- Applegate, D., Bixby,R., Chvatal V., Cook,W. (1998). *"On the solution of the Traveling Salesman Problem"*. DocumentaMathematica-Extra Volume ICM III. 645-656.

- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, USA.

- Clay Mathematics Institute. (2000). *Millenium problems*. www.claymath.org/millenium/

- Christofides, N. (1976). *Worst-case analysis of a New Heuristic for the Traveling Salesman Problem*. Report 388, Graduate school of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

- Climer, S., Zhang, W. (2004). *Take a walk and cluster genes: A TSP-based approach to optimal rearrangement clustering*. 21st International Conference on Machine Learning (ICML'04), Banff, Alberta, Canada, 169-176.

- Cook, J. William (2012). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. New Jersey: Princeton University Press.

- Dantzig, G., R. Fulkerson y S. Johnson (1954). *Solution of large-scale traveling salesman problem*. The Rand Corporation, Santa Mónica, California.

- Fuentes-Penna Alejandro and González-Ramírez Marcos S. (2013). *Meta-Heuristic Algorithm based on Ant Colony Optimization Algorithm and Project Scheduling Problem (PSP) for the Traveling Salesman Problem*. Journal of Network and Innovative Computing ISSN 2160-2174 Volume 1 (2013) pp. XXX-XXX © MIR Labs, www.mirlabs.net/jnic/index.html.

- García, M. (2014). *Problema del viajante de comercio (TSP), Métodos exactos de resolución*. Universidad de La Laguna.

- Gilmore, P.C., Gomory, R.E. (1964). *Sequencing a one state-variable machine: A solvable case of the traveling salesman problem*. Operations Research 12, 655-679.

- Glover, F., Gutin, G., Yeo, A., Zverovich, A. *"Construction Heuristics for the asymmetric TSP"*. European Journal of Operational Research 129 III. 2001. 555- 568.

- Held, M., Karp, R.M. (1962). *A dynamic programming approach to sequencing problems*. Journal of the Society of Industrial an Applied Mathematics 10, 196-210.

- Kawamura, H., Yamamoto, M., Suzuki, K., Ohuchi, A. (1998). *Cooperative search on pheromone communication for vehicle routing problems*. IEEE Transactions on Fundamentals, E81-A: 1089-1096.
- Okamoto, Y. (2004). *Traveling sakesman games with the Monge property*. Discrete Applied Mathematics 138, 349-369.
- Stockdale, M. (2011). *El problema del viajante: un algoritmo heurístico y una aplicación*. Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales. Departamento de matemáticas.

CONCLUSIONES

- Se constuyó una reseña histórica por medio de la cual es posible entender aspectos relacionados con el génesis, desarrollo y el nivel de aplicación que tiene actualmente la Investigación de Operaciones, y en particular de los problemas de Optimización Combinatoria.
- Se realizó una revisión del estado del arte de las técnicas de solución utilizadas para resolver los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y el Travelling Salesman Problem.
- Se planteó la formulación matemática de los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y el Travelling Salesman Problem.
- Se implementaron algoritmos exactos y heurísticos (Búsqueda exahaustiva, Programación Dinámica, Programación Lineal Entera, Branch and Bound, Dijkstra, Floyd-Warshall, Bellman-Ford, heurísticos del coeficiente de rendimiento, heurístico de corte, el heurístico del vencino más cercano) para tratar los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y el Travelling Salesman Problem.
- Se implementó la herramienta computacional (WinQsb) para resolver algunos problemas de Optimización Combinatoria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y el Travelling Salesman Problem.
- Se diseñó una metodología para tratar los problemas de Optimización Combinatoria, dicha metodología dista de la forma en que normalmente los libros clásicos tratan estos temas, ya que los problemas se resuelven de una forma intuitiva, sencilla y práctica, la cual fue estructurada de la siguiente manera: a) Introducción; b) Formulación Matemática del problema; c) Tratamiento metodológico; d) Problemas propuestos; e) Problemas resueltos y f) Bibliografía.

PERSPECTIVAS DE FUTURO

- Realizar la revisión del estado del arte de las técnicas utilizadas para resolver otros problemas de Optimización Combintaria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.
- Plantear la formulación matemática de otros problemas de Optimización Combintaria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.
- Implementar algoritmos exactos y heurísticos para resolver otros problemas de Optimización Combintaria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.
- Implementar la herramienta computacional WinQsb para resolver los problemas de Optimización Combintaria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.
- Implementar la metodología propuesta para abordar otros problemas de Optimización Combinatoria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.